NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

IMPROVING SOFTWARE QUALITY AND MANAGEMENT THROUGH USE OF SERVICE LEVEL AGREEMENTS

by

Leonard T. Gaines

March 2005

Dissertation Supervisor James Bret Michael

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2005	**3. REPORT TYPE AND DATES COVERED** Dissertation	
4. TITLE AND SUBTITLE: Improving Software Quality and Management Through Use of Service Level Agreements			**5. FUNDING NUMBERS**
6. AUTHOR: Leonard Troy Gaines			
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) Naval Supply Systems Command Mechanicsburg, PA			**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)***

In this dissertation we explore the use of service level agreements (SLAs) to improve the quality and management of software-intensive systems. SLAs are typically used in outsourcing contracts for post-production support. We propose that SLAs be used in software acquisition to support quality and process control throughout the lifecycle (requirements engineering through post-production support) of a software-intensive system. The hypothesis was tested using two methodologies. The first method explained how SLAs could be used throughout a system's lifecycle to improve software quality. This concept was validated by a survey of IT professionals. The results of the survey indicated that practitioners in the IT field felt that SLAs could be used to improve overall quality in the development effort and in the end product. The second approach was to develop actual SLAs for a specific lifecycle phase (post-production) to illustrate the concepts of SLAs and to demonstrate their value as a quality control and management tool.

14. SUBJECT TERMS Software Engineering, Software Metrics, Software Management, Software Acquisition, and Service Level Agreements			**15. NUMBER OF PAGES:** 434
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION 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

Approved for public release; distribution is unlimited.

IMPROVING SOFTWARE QUALITY AND MANAGEMENT THROUGH THE USE OF SERVICE LEVEL AGREEMENTS

Leonard T. Gaines
Commander, United States Navy
B.S., University of Nevada, 1986
M.S., Naval Postgraduate School, 2000
M.S., Industrial College of the Armed Forces, 2004

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN SOFTWARE ENGINEERING
from the

NAVAL POSTGRADUATE SCHOOL
March 2005

Author: _____
Leonard T. Gaines

Approved by:

_____	_____
Bret Michael	Dan Boger
Professor of Computer Science	Chairman of Information Sciences
Dissertation Supervisor	
Committee Chairman	
_____	_____
John Osmundson	Man-Tak Shing
Professor of Information Sciences	Professor of Computer Science
_____	
Rex Buddenberg	
Senior Lecturer Information Sciences	

Approved by: _____
Peter Denning, Chairman, Department of Computer Science

Approved by: _____
Julie Filizetti, Associate Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this dissertation we explore the use of service level agreements (SLAs) to improve the quality and management of software-intensive systems. SLAs are typically used in outsourcing contracts for post-production support. We propose that SLAs be used in software acquisition to support quality and process control throughout the lifecycle (requirements engineering through post-production support) of a software-intensive system. The hypothesis was tested using two methodologies. The first method explained how SLAs could be used throughout a system's lifecycle to improve software quality. This concept was validated by a survey of IT professionals. The results of the survey indicated that practitioners in the IT field felt that SLAs could be used to improve overall quality in the development effort and in the end product. The second approach was to develop actual SLAs for a specific lifecycle phase (post-production) to illustrate the concepts of SLAs and to demonstrate their value as a quality control and management tool.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my family, friends, and co-workers who supported me in this effort. I would especially like to thank Dr. Bret Michael for his encouragement, hard work and help in bringing this dissertation to fruition. I appreciate the amount of time and effort it takes to review and provide comments on a dissertation of this size. Given their extremely busy schedules, I sincerely appreciate the efforts and dedication of Dr. Dan Boger, Dr. Man-Tak Shing, Dr. John Osmundson, and Professor Rex Buddenberg. The author would also like to acknowledge Scott Price and Joseph Vickery from EDS for their support with drafting the SOW in Appendix A. Finally, I would like to thank my wife, Sally, for her patience, support, and sacrifice.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Management and end-users have become increasingly dependent upon software-intensive systems to support new ways of conducting business. These critical software-intensive systems are becoming more complex, and difficult to manage, yet the performance and quality expectations from management and the end-users continue to increase. Unfortunately, despite software's increased importance to organizations, the quality of software can be lacking.

The dissertation describes a new approach to software acquisition: application of service level agreements (SLAs) throughout a system's lifecycle and at each major phase of software development and maintenance to improve the overall quality of the end product. The hypothesis is that the use of the SLAs in the software acquisition process can improve product, process, project, and post-production quality by identifying and defining relevant quality factors, quality metrics, quality thresholds, methods of measurement, and by establishing penalties for failure to meet quality requirements.

The basis for the hypothesis is our theory that the SLA development process aids requirements engineering by identifying software quality factors that support the critical business processes the software development or maintenance project supports. The quality factors that are addressed in the SLAs then drive architectural and design decisions about the business-critical system. If developers and maintainers of business-critical systems know which of the characteristics are most critical to project success, they can select – within the constraints of time and budget – among system architecture, design, and implementation alternatives that have a high likelihood of meeting the quality goals set forth by the stakeholders for the end product.

To test the hypothesis, we used two approaches. The first approach explained how SLAs could be used throughout a system's lifecycle to improve software quality. This approach was validated by a survey of information technology (IT) professionals. The second approach was to develop actual SLAs for a specific lifecycle phase (post-production) to illustrate the concepts of SLAs and to demonstrate their value as a quality control and management tool.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. EXECUTIVE OVERVIEW

In the past, the typical information system tended to be homogeneous and stovepiped, and was typically developed from scratch by a small number of vendors. In contrast, today's typical information system is software-intensive and distributed, composed of heterogeneous subsystems, supplied by numerous vendors. The subsystems themselves can consist of a mix of legacy and new-system development. Software is viewed by many as the means for making systems readily adaptable to change, as the environments in which the systems operate change. In this dissertation we treat the topic of service level agreements (SLA) in the context of their use in managing modern information systems over their entire system lifecycle.

Many of the advances in the principles and mechanics of software engineering provide the software engineer with a means for improving the quality of software-intensive information systems. However, actual practice does not always take advantage of these advances. This can be attributed to such factors as training problems, the rush-to-market mentality, and lack of proper quality control throughout the lifecycle of the information system. Although quality control is the responsibility of the program manager, he or she may choose to defer addressing quality to later in the system lifecycle, focusing initially on realizing functional system requirements. As experience has shown, retrofitting quality—and non-functional requirements in general—into an information system can be difficult to do technically or even cost-prohibitive to achieve in some cases. In this dissertation, we argue that program managers must address software quality through the system lifecycle and that SLAs provide a means for managing the activities needed to build quality into software-intensive information systems.

A SLA is a contractual mechanism that defines quantifiable quality metrics, acceptable service levels, the method of measurement, responsibilities of the parties to the SLA, and incentives—both positive and negative reinforcements—for meeting agreed upon service levels. A SLA can be used for in-house development efforts and services as well as those that are outsourced. Further, a SLA can also be used in any stage of the

application's lifecycle. Typically, a set—something akin to a portfolio—of SLAs are used in conjunction with the management of an information system, with each SLA representing a distinct quality attribute (e.g., reliability, maintainability) of the information system or dimension of system development and maintenance (e.g., product quality, process quality, project quality, and quality in post-production maintenance and services).

1. Hypothesis

Service level agreements can improve the management and quality of software-intensive information systems throughout the system's lifecycle. Embedded software and other specialized application of software are not within the scope of this paper.

2. Methodology

To test the hypothesis, we utilized two approaches. In the first approach we explained how SLAs could be used as a quality control tool in the various phases of software development to improve product, process, project and post-production quality. Our research identified areas within the development and post-production effort where SLAs could be effectively utilized, as well as provided examples of standards and quality models that could be incorporated into SLAs. We validated these concepts through a survey instrument administered to information technology (IT) professionals.

The questionnaire consisted of three sections. The first section provided the subject with a brief introduction to the topic of software quality, and a short discussion of how SLAs can contribute to software program management from conceptualization of an information system through post-production support. The second section was a case study illustrating a real-world scenario along with a SLA for availability. The last section consisted of a questionnaire comprised of twenty-nine questions and a comment section. Each statement had a corresponding Likert scale from one to five, with a one representing strong disagreement and a five indicating strong agreement. The survey was conducted from a web site.

The second approach was to develop SLAs for a specific phase of the software lifecycle to further illustrate how SLAs can be used as a quality control tool and to demonstrate the usefulness of the new SLA format. Although the SLAs that were

developed apply to post-production support, a similar approach can be utilized to apply quality factors to other phases of development. The SLAs were also created to demonstrated how SLAs could be used as a template for requirements elicitation and to show that they can and should be tailored to meet project specific needs.

3. Results

The survey supported the hypothesis that SLAs can improve the management and quality of IT intensive systems throughout their lifecycle. Twenty-two of the twenty-five statements had a statistically significant difference from the null hypothesis (mean equal to three on the Likert scale, which indicates a neutral feeling about the statement).

4. Original Contribution

This dissertation has three major original contributions to the field of software engineering. The first contribution is a unique approach to improving software quality from a software acquisition perspective. For numerous reasons software acquisition tends to concentrate on the functional aspects of an information-intensive system. Unfortunately, this approach often leads to poor software quality as software can be of poor design, but still meet functional requirements. In an effort to improve software quality this dissertation advocates the use of SLAs in software acquisition contracts to specify performance-based requirements relating to product, process, project and post-production quality. This dissertation demonstrates how SLAs can be applied to the entire lifecycle of a software-intensive system in an effort to improve the quality of the management and development of the system. SLAs are not a new concept, however they are used primarily in post-production support. In this dissertation we take the concept of SLAs and demonstrate how they can be used as a quality control and management tool throughout the software development cycle (i.e., requirements, design, coding, testing, post-production support).

The dissertation introduces a unique format for the SLAs. The format forces the SLA development group to define in detail (e.g., in terms that all stakeholders understand) the services to be performed, quantitative service levels, the method of measurement, and time frames or periodicity of measurements. The format helps to ensure that all parties understand the terms of the SLAs by stating the responsibilities of

the contractor and program manager, stating assumptions, deliverables, stating who will perform monitoring, and how monitoring will be performed. The format also ties the quality requirements to specific business needs and stakeholder concerns. Providing the rationale for measuring the service ensures the development team has considered whether the service and quality thresholds are relevant to business needs, that the quality thresholds are realistic, and that metrics are meaningful and provide value.

Although SLAs are sometimes found in contracts with External Service Providers (ESPs) for post-production support they are often used more to set expectations rather than establish quality control measures. They are often poorly defined, they lack information concerning monitoring techniques, and they generally favor the ESP.

In this dissertation we have developed thirteen original post-production SLAs that are far more extensive than those found in the research conducted. The SLAs in appendix (A) were developed to illustrate how SLAs can be used as a quality control tool, not just for post-production, but for the other phases of the software lifecycle as well. The SLAs in appendix (A) were used in actual source selection negotiations with very favorable results.

5. Expanding the Body of Knowledge

Although numerous software engineering disciplines are discussed, this dissertation has made contributions to the body of knowledge in the disciplines of software acquisition, requirements analysis, software quality and software project management.

There is currently a lack of theoretical basis or intellectual body of knowledge in the field of software acquisition. Although there is a great deal of research concerning software development methods and their affect on project success in terms of cost and schedule, similar research on contracting methodology for software development is lacking. This dissertation proposes a methodology for acquiring software that focuses on project, process, product and post-production quality. This approach goes beyond traditional acquisition by applying a holistic view of quality throughout a system's lifecycle. SLAs can be used as a quality control tool to enhance other software acquisition approaches such as the Software Acquisition Capability Maturity Model (SA-

CMM) (Software Engineering Institute Mar 2000), IEEE Recommended Practice for Software Acquisitions (IEEE Std. 1062) or Performance-Based Acquisition. (DoD USD (A&T) Utilizing SLAs in the acquisition process is an attempt to correct many of the software acquisition deficiencies sited in numerous articles, studies, and General Accounting Office (GAO) reports. Although this dissertation does not empirically demonstrate that SLAs will lead to project success and better quality, it does provide a foundation upon which future studies can be based.

The SLA development process supports and incorporates many of the theories proposed in the field of software requirements engineering such as Facilitated Application Specification Technique (FAST) (Zahniser), Mizuno and Akao's Quality Function Deployment (Zultner, Krogstie) as well as use cases and scenarios (Hickley, Sutcliffe). Many of the requirements elicitation techniques proposed by practitioners and academia can be incorporated in the SLA development process to generate quality requirements. For example, the SLAs presented in appendix (A) can be utilized in scenario elicitation.

SLAs also enhance existing requirement engineering techniques or methods. SLAs concentrate on non-functional quality requirements, which are not always considered in other methods. Due to the nature of contracting for software services, SLAs introduce quality requirements early in the lifecycle where they are most effective. Quality software requires more than just identifying quality requirements. Monitoring and measuring the requirements is necessary to ensure the requirements are being met. The literature on software requirements almost always implies that just because requirements are specified, that they are incorporated into the final product. This is rarely the case. SLAs enhance existing software requirement techniques by instituting a measuring and monitoring philosophy (quality control) and enforcing requirements by use of penalties for non-compliance.

The SLA development process also enhances traditional software requirement techniques. The level of detail necessary to develop the SLAs requires an understanding of the business processes the system is supporting, it incorporates multiple perspectives, and it requires a prioritization of the quality factors chosen. The development effort will

not only generate discussion on which quality attributes are appropriate for the software system, but it will also identify whether resources, employee skill sets, and management support exist to properly support and enforce the SLAs. SLAs and specifically the format proposed in this dissertation will help to produce requirements that are quantifiable, measurable, meaningful, and support business processes.

This dissertation adds to work that has been conducted on software quality. While this dissertation does not introduce a new model for software quality, or a new measurement of quality, it does introduce the use of SLAs as a means to contract for software quality over the lifecycle of a product. SLAs are the practical implementation of many of the quality models that will be discussed later in the dissertation.

The SLA development effort also contributes to the discipline of software quality by incorporating quality (functional and non-functional) requirements in the requirements engineering process. The development effort evaluates many of the quality models and metrics proposed in literature. These metrics and models are then applied in part or in whole to measure or specify process, product, project, and post-production quality.

There is no single quality model that can extend through the entire lifecycle of a software product. SLAs are a means to incorporate many quality factors and models simultaneously to best support the system throughout its lifecycle. The SLA development effort involves an analysis of the various quality factors and models to determine which best support the system given performance expectations, budget and time constraints, and the purpose of the system. Prioritizing the quality factors and resolving quality requirement conflict are an important part of the development process. It is very likely that multiple quality models will have to be incorporated to evaluate the deliverables at the various stages of the development cycle.

This dissertation has also added to the body of knowledge related to software project management. SLAs enhance many of the existing processes or models associated with software project management such as Performance-based Management (Plunkett), Software-Performance Engineering (Smith, C. 1988), and Capability Maturity Model Integration (CMMI) (Software Engineering Institute, Aug 2002) by instituting the software quality control measures that are implied in these models.

In addition to quality management and quality control, SLAs can assist program managers in many of the tasks identified in project management models as important to the success of the project. In the SLA development effort, the project is scoped, risks are identified and analyzed, resources are evaluated, quality factors are prioritized, specific business needs are identified, and success factors for those business needs are defined. SLAs also help the program manager in the areas of financial management, customer relations, configuration management, and especially contract management.

6. Outline of Dissertation

Chapter I outlines the importance of IT systems, and describes the difficulty that both public and private sectors have had in developing quality IT systems. The chapter also provides a detailed discussion on software quality. Chapter II defines SLAs, discusses how they are utilized, and describes a recommended format. Chapter III outlines an 8-step process for developing SLAs and provides a case study describing how the SLAs in appendix (A) were developed. Chapter IV provides a detailed discussion on software development models, illustrating how SLAs can support various approaches. Chapter V describes how the SLA development process can support and enhance many of the recommended requirements engineering processes and techniques. Chapter VI discusses how the quality metrics and quality factors incorporated in the SLAs can influence the architecture and design of the system. Chapter VII illustrates the importance of selecting the appropriate software quality factors to incorporate in the SLAs. Chapter VIII describes how SLAs can be utilized as a quality control tool to assist the program manager in managing the configuration of the project. Chapter IX explains how SLAs can also assist the program manager with many aspects of program management and oversight. Chapter X is a detailed discussion on the research methodology and results. Chapter XI contains the conclusion and makes recommendations for future work. Appendix (A) is a statement of work (SOW) along with thirteen SLAs that were used in a proposal for post-production support. Appendix (B) contains the survey instrument. The final section is Appendix (C), which provides a breakdown of the results of the survey.

7. Deliverable

The concepts discussed in this dissertation were applied in the development of the SLAs and the SOW found in appendix (A). The SLAs and SOW in appendix (A) were developed for the post-production hosting services under CLIN 0029 of the Navy/Marine Corps Intranet (NMCI) contract. The SLAs and SOW were designed to allow program managers to select from three levels of services to support their programs. Programs needing more advanced services would be able to modify the CLIN to support their needs. The CLIN 0029 SOW and SLAs are currently still in contract negotiation.

B. IMPORTANCE OF INFORMATION TECHNOLOGY

IT has offered an unprecedented opportunity for organizations to improve the efficiency and effectiveness of its operation. The rapid growth of the Internet has lead to an ever increasing reliance by organizations on interconnected computer systems to provide critical operational services, from business processes to coordinating decentralized command, control, computers, communications, intelligence, sensors and reconnaissance (C4ISR) systems.

One can argue that IT-based systems have become the most critical, multi-faceted strategic tool any business or organization possesses. (Info Tech) Organizations that have properly integrated IT into their overall business processes and have invested in the most current infrastructure have a significant advantage over any competition that has not taken advantage of IT.

The reliance on IT systems to provide strategic and tactical advantages has placed ever-increasing levels of pressure on the IT department to provide quality services and products than ever before. Interruptions to IT systems are having a far greater impact than before in terms of opportunity loss, revenue loss, customer dissatisfaction, and efficiency. As managers realize that their mission-critical processes are tied to IT services, they are demanding more control over the quality of the services provided.

Another factor bringing IT quality to the attention of senior management are the various third-party vendors, system integrators, or external service providers (ESP) that market IT services that are similar to those offered in most IT departments. Outsourcing

is forcing IT organizations to reevaluate their relevancy to the organization. When top executives hear the sales pitches from ESPs, they expect similar or higher levels of service from their internal staffs. Competition has started to drive the levels of service higher and higher, especially when service performance is really the only differentiator the ESPs have with one another.

Information flow is the lifeblood of an organization allowing it to enable its personnel, respond to customers, and react to the external environment. An organization's ability to gather, manage, and use information will determine its success. (Gates) Leveraging information technology allows organizations to interconnect disparate processes and information that was separated logically, physically, and chronologically. The rapid growth of technology along with the greater globalization of enterprises has brought IT management to "center stage". However, as information systems become more complex and distributed, they also in general become increasingly difficult to manage, yet the performance expectations for the system, from management, and the end-users continue to increase.

All organizations want world-class quality levels, but achieving those quality levels requires a holistic view of quality that incorporates leadership support, repeatable and measurable quality processes and controls, resource planning, vision, customer support, and service-level management. Organizations must do more than identify and incorporate quality attributes in their requirements, they must also monitor quality metrics to ensure those quality requirements are being met. Quality is not something that is inherent in the development process: it must be planned, monitored and incorporated as part of standard business practices.

C. SOFTWARE QUALITY

There are numerous definitions of quality. The ISO 9000 model defines quality as the degree to which a set of inherent characteristics fulfills requirements. (Tricker) ISO 9126, a refinement of the ISO 9000 model, which proposes a quality standard for software product evaluation, defines software quality as the totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied

needs. (Hansen) Pressman states that software quality is conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software. (Pressman)

It is interesting that both ISO model definitions and Pressman's definition are based on an assumption that all stakeholders have an input into the requirements-specification process. An IT system may meet all of the program requirements and thus be viewed as being a quality product. However, the IT system will not be perceived as a quality product if the product does not perform according to the end-user's perspective. Many believe that quality is based upon the perceptions of the stakeholders. This view is also supported by Garvin, who stated that quality is multifaceted and can be viewed by many perspectives. (Garvin) However, it is generally recognized that the consumer of the product is the ultimate judge of a product's quality. (Glass, Tice, Briones, Weigers) The IEEE standard 610-1990 does incorporate user needs, by defining software quality as the degree to which a system, component, or process meets specified requirements and meets customer or user needs and expectations. (Schmidt)

Software quality can be broken down into four areas of focus. The first area, product quality, is concerned with the requirements and specifications of the product as it applies to the attributes or characteristics of the software product. This area could also be referred to as end-product quality. The second area, project quality, is concerned with the metrics and measurements associated with the software production effort. (Wheeler, Hilburn) The third area is process or management quality, which is concerned with the processes, planning and controls used to develop and manage the software product. The last area of focus is on post-production quality or deployed application management. Although there is some overlap with process quality, this last area is focused on software maintenance, IT system performance, and hosting services after the application has been placed into production. Software quality models have been developed in all of these areas in an attempt to evaluate and/or predict software quality.

1. Product Quality

Quality attributes are generally used to describe the degree to which software possesses certain characteristics. Quality can be viewed from numerous perspectives, and certain attributes are more preferable to others depending on the objective of the IT system. As such, numerous quality attributes have been identified. When referring to product quality, two perspectives are generally represented: those of the user and those of the developer.

In addition to the functional aspects of a system, the end user wants the product to exhibit specific qualities that will assist them in performing their task. From a user's perspective, some of the common quality attributes used in the quality models include availability, usability, integrity, interoperability, and reliability. Personnel involved in the development of software or its maintenance may be more concerned with the software attributes such as portability, testability, maintainability, and reusability. (Wiegers)

Product quality models concerned with the developer's perspective can be further broken down into three categories. The first category is concerned with those quality factors, or their associated quality metrics that involve attributes associated with the software code. A common quality metric for software code is defects per thousand lines-of-code (KLOC). The next category is concerned with quality metrics associated with the structure or architecture of the software. Structure quality metrics are concerned with the features, components and relationships among the components. Common structure quality metrics are quantitative counts of the sources (fan-in) and destinations (fan-out). The last group contains hybrid quality metrics, which combines code and architecture quality factors. An example would be evaluating complexity by analyzing or weighing against the lines-of-codes in the modules. (Kafura)

One of the first software quality models to address product quality was the McCall quality model, based on earlier work by Boehm (Boehm). McCall's model consists of a number of questions and a subjective grading criterion based on a Lickert scale from 0 to 10. McCall defined quality in a hierarchical manner in which quality factors defined a key characteristic of the software, such as 'maintainability'. Quality

factors consisted of quality criteria that represented an attribute of the quality factors, such as 'understandability.' Finally, quality metrics were used to assign quantitative measurements to the quality factors. (Pressman, Ward, Kafura)

There are numerous software product quality models that incorporate software quality factors or metrics in an effort to benchmark or measure software quality. Some of the better-known quality models include early work done by Halstead, who calculated complexity based on the number of operators and operands. (Ogasawara) The ISO 9126-1 quality model is also well known. The ISO 9126-1 model incorporates the quality factors functionality, reliability, usability, efficiency, maintainability and portability. (Cross, Ward) The Hewlett-Packard FURPS model is also well known. (Pressman)

There are also numerous software quality models that concentrate on specific quality factors such as complexity. (Ogasawara, McClure) In his book Software Complexity- Measures and Methods, Horst Zuse identifies over ninety models for describing the software attribute complexity. Other quality models are specific to object-oriented systems (Coppick, Pritchett), some are specific to a language (Pritchett) or COTS components (Bertoa, Hansen), while others are only applicable at run-time. (Bass)

2. Project Quality

Project quality is concerned with metrics that allow an organization to manage, track, and improve the quality of the software-development effort. One of the most common quality factors involving project quality is project estimation. Project estimation models such as COCOMO II (Boehm), Albrecht's Function Points (Albrecht, Jones), and Putnam's Software Life-cycle Model (SLIM) (Putnam, Chulani) address the cost to produce software, errors or defects that can be expected, as well as the level of effort required to produce the software.

Some of the project quality metrics that Motorola used to measure their software-development projects included software-defect density, adherence to schedule, estimation accuracy, reliability, requirements tracking, and fault-type tracking. (Daskalantonakis)

Other project quality models such as DoD Std 7935 are concerned with the degree of formalism necessary to manage the project. (McConnell) Another metric used in assessing project quality is risk, which can be defined as any variable within a project that

results in project failure. General risk areas are schedule risk, requirements risk, budget risks and personnel risk. (Padayachee) There are a number of risk assessment models including Gilb's risk heuristics (Gilb), Boehm's classification of risk (Boehm), Keil's follow on identification of risk factors (Keil), the USAF AFCS/AFLC Pamphlet 800-45 which outlines software risk identification and abatement (Pressman), interpretivist approaches (Gemmer, Padayachee), risks associated with enterprise software projects (Charette, Sumner), and Noguiera's risk assessment model. (Noguiera de Leon)

3. Process Quality

Quality metrics also apply to the processes and business practices used to manage software throughout its lifecycle. Quality in the context of software process management refers to an adherence with explicit process requirements and those implicit processes necessary to meet user requirements and produce quality software. Process metrics allow a holistic view of the activities that organizations are taking to ensure a quality software product. Processes provide a clear understanding of what an organization does and the quality controls it has in place to do those activities. (Tricker)

There are many who believe that the quality of the development process is the best predictor of software product quality. (Fenton) Repeatable software processes such as the Software Engineering Institutes Software Capability Maturity Model for software (SW-CMM), which lists five levels of organizational maturity levels, and the International Standards Organization (ISO 9001:2000) are designed to improve software quality, productivity, predictability and time to market. (Paulk, McGuire) There is also some empirical evidence that there is a correlation between process maturity and software quality. (Harter, Diaz, Ferguson)

Other models of process quality include the new Capability Maturity Model Integration (CMMI) model. CMMI integrates 3 CMM models into one to eliminate problems with different architecture, semantics, and approaches. (SEI) Humphery developed the personal software process (PSP) to assist software engineers in producing quality software. (Humphrey) Other process models include cleanroom engineering that has shown reduced errors per KLOC for small projects (Fenton), and the quality management metric (QMM) (Machniak, Osmundson). There are also numerous IEEE

and ISO standards that provide processes on everything from software engineering product evaluation (ISO/IEC 14598) to selecting appropriate quality metrics (IEEE Std. 1061-1998).

4. Post-Production Quality

Quality control does not stop once a software product has been deployed. Quality factors still need to be applied to the application performance, maintenance efforts, and hosting services throughout its lifecycle. Monitoring the performance of the application once it is deployed is essential in quality control and maintaining customer satisfaction. Much of the application performance monitoring in the initial phases of deployment is used to validate product-quality factors identified in the initial requirements. However, in the post-production environment there is also an emphasis on monitoring system performance in terms of resource utilization, system capacity, network utilization and quality of service, storage management, and security.

Many of the quality models involving deployed applications are concerned with software maintenance and the quality factors that make maintenance cheaper and more effective. Some of the maintenance-quality factors deal with ease of change (Royce), others deal with architectural design to promote maintenance (Hulse, Garlan), defect management (Kajko-Mattsson), organizational structure (Briand), complexity (Banker), and change management. (Bennett)

Quality factors with deployed software are also concerned with the IT system as a whole. Quality is not just concerned with the application itself: it is also concerned with the IT system as a whole, across distributed components. Part of that distributed system is the network. There are numerous quality metrics that can be applied to network quality of service. (Clark, Tanenbaum, Lee, Hochstetler, Packeteer) Quality metrics are also applied to the host server. Quality metrics such as application-resource utilization (Aries), bandwidth utilization (Eager), concurrent user management (Aweya), and server performance (Dalal, Gama) are also utilized to address system-level quality. Hosting services are another area that needs to be addressed when discussing the quality of production software. Traditional hosting metrics have centered on total cost of operation

(TCO) benchmarking, and help desk support metrics, but areas such as backups, storage, configuration management, and security also need to be addressed.

There are numerous software-quality models and metrics that can be incorporated into SLAs. The models or quality factors chosen will depend on those quality attributes that best support the underlying business process. Regardless of the software-quality models incorporated in the SLAs, the software metrics must be meaningful, quantitative, and measurable.

In this dissertation, the term quality is used loosely to describe the degree to which a system, component, or process meets specified requirements and meets customer or user needs and expectations. Quality thresholds or quality metrics are those measurements that specify the quality factors or quality requirements.

D. CHALLENGES IN OBTAINING QUALITY SOFTWARE

The software program manager is responsible for evaluating the program requirements and determining the methodology or process to deliver and maintain quality software. There have been a number of initiatives proposed to improve the quality of software through its lifecycle. Most approaches are based on the tenet that quality must be designed into a product. Approaches such as formalizing specifications (Berzins), use of development standards and models, and utilizing architecture for quality analysis support this approach. These approaches can be supplemented, for instance, by using programming languages such as Ada that are designed to prevent common design and coding errors, or utilizing rigorous testing and third-party debugging tools.

If there are numerous approaches to developing quality software, why are there still problems? Part of the answer lies with the lack of meaningful dialog between the developers, end-users and management. Unrealistic completion dates, requirements churn, poor requirements elicitation, and lack of proper resources all lead to development problems. Additionally, just because standards exist for developing software does not mean that they are being used. In many cases adherence to developmental standards

requires additional training, additional development time, additional funds and a commitment from upper level management that those standards will be inspected and enforced.

In his book "Decline and Fall of the American Programmer," Yourdon estimates that eighty-five percent of US software organizations operated at level 1 of the SW-CMM. (Yourdon) This fact was reemphasized by Dietz who stated that most of the software companies that he evaluated were at level 1 of the CMM. (Dietz)

A study published by the Standish Group reveals that the number of software projects that fail has dropped from 40% in 1997 to 26% in 1999. However, the percentage of projects with cost and schedule overruns rose from 33% in 1997 to 46% in 1999 (Noguiera) In another Standish study in 1999, a survey of 1,500 software projects found that 31% of the projects were canceled and of those projects that were delivered on average only 61% of the originally specified features were delivered. (Cross)

Despite software's increased importance to organizations, software program managers have not improved the quality of software. (Anthes) There are numerous examples of software errors leading to major incidents, including the Denver airport baggage handling system, the Hershey Foods ERP implementation, the Toys-R-Us e-commerce site continuing to promise delivery of Christmas gifts after shipping cut-off dates, and the USS Yorktown Smart Ship system failure. (Slabodkin, Huckle)

In the article "Why Software is so Bad", Mann offers a number of reasons why the quality of software tends to be poor. Mann states that software quality is actually getting worse rather than better, despite the advances in software engineering theory, processes, methodology and tools. Poor software quality can be attributed to the following:

- The perceived need to hurriedly develop and market a software-based product to be the first to market; such an approach can result in software artifacts that contain software flaws and are difficult to test and maintain. In a 60-day development cycle, which is not uncommon, programmers are not going to spend two weeks searching for a bug, despite risks associated with deploying a faulty product. (Blacharski)

- Software can be poorly designed. This is due in part to the poor training programmers have received, and the fact that as programmers bounce code off of the complier to fix errors, they often deviate from the original designs and end up with sloppy, poorly documented code.
- Testing software often requires a different skill set than programming. Often the testing personnel are not properly trained, or are not given the time to test properly. Too many organizations are relying on testing as the primary means to improve quality instead of designing the application with quality factors built into their initial requirements—the latter approach actually can improve our ability to test systems.
- Software is not designed for testing. The designers do not utilize good component level design or software architecture, the software's modularity and corresponding interconnectivity is not well defined, and the application is not internally coded to throw exceptions, or write faults to a log.
- Software fails to meet the customer's expectations. The software developer must looks at requirements from the user's perspective, the business' perspective, and the programmer's perspective. Too often the user is not a part of the requirement elicitation process.
- Requirements churn contributes to the poor reliability of software, as designs are altered, interfaces added, unplanned modules are glued together, with little consideration given to the additional resource demands.
- Post-production support plays a large role in the success of an application, but software developers do not normally address it in their planning.
- The application needs to be hosted in an environment that supports the application's functionality. Software quality can be adversely affected by lack of resources within the server, and by network and bandwidth constraints.
- Maintaining software without proper documentation or configuration information is very difficult and expensive. Additionally, without proper documentation it is difficult to compare the original requirement specifications to the product throughout the software's lifecycle.

E. QUALITY PROBLEMS IN THE DEPARTMENT OF DEFENSE

The next three sections discuss some of the problems that the DoD has with the management of software-intensive information systems, recruiting and retaining competent IT personnel and outsourcing. Although these sections focus on the DoD, many of the same problems can be found in the commercial sector.

In the past, the DoD has not excelled at managing software-intensive information systems through their lifecycles. Managing information systems can be challenging. Utilizing the latest technology to exploit information requires highly developed intellectual and managerial skills, which are rare attributes (Rocheleau). The difficulty in managing these systems has been demonstrated by the numerous system development and maintenance projects within the DoD that lacked sound planning, had poor controls, lacked measurements for success, and did not meet expectations.

From 1986 to 1996, the US Government spent 200 billion dollars on information technology that did not produce the results that were desired. (Deputy Assistant Secretary of Defense) One example is the Corporate Information Management (CIM) initiative. In October 1989, the DoD attempted to improve and consolidate almost 2,000 information systems relating to transportation, depot maintenance and material maintenance. By October 1993, the DoD determined that efforts to develop and complete these logistics systems would take too long to develop and would not produce the costs savings they initially anticipated. In response, the DoD standardized on its best logistics information systems—in terms of performance, maintainability, and other measures of effectiveness—across all military services. This “migration strategy” as it was termed, was designed to quickly produce cost savings. By 1995, the DoD realized that its migration strategy for materiel management and depot maintenance consumed more resources than it had anticipated, took longer than expected, and did not produce the benefits expected. Over 700 million dollars was spent migrating material management systems before abandoning the project, having failed to produce a single operational system. (U.S. GAO OCG-99-4) The CIM and migration-strategy effort cost eighteen billion dollars without achieving its objective. The DoD abandoned its efforts at

standardizing the systems and opted instead to try to achieve interoperability between the different services' information systems, and privatize some functions. (U.S. GAO AIMD-96-109)

Despite the failures of the CIM and the migration strategy, the US General Accounting Office (GAO) noted that the interoperability and privatization approach suffered from the same managerial problems that plagued the two prior attempts at system consolidation. The DoD did not even conduct a thorough cost-benefit study to determine if the new strategy would achieve a positive return on investment. The DoD failed to tie its efforts to its overall business objectives using strategic planning. It had also not adequately explored better commercial alternatives such as reengineering or outsourcing. (U.S. GAO AIMD-97-6, U.S. GAO 01-244)

1. Clinger-Cohen Act

On October 12, 1994, then Senator Cohen of Maine and a member of the Senate Governmental Affairs Committee released a report entitled "Computer Chaos: Billions Wasted Buying Federal Computer Systems." The report was a summary of reports from the GAO and Inspector General (IG) that detailed problems with major software-development projects that were in progress. The report concluded that antiquated systems were costing the government billions of dollars, government-planning efforts were inadequate, and the acquisition process forced the government to pay more for less. (Peckinpaugh)

The Information Technology Management Reform Act (ITMRA) of 1996 coupled with the Federal Acquisition Reform Act became known as the Clinger-Cohen Act. Congress' intent in passing the Act was to solve some of the longstanding problems associated with the acquisition and maintenance of information systems by the DoD. Among those problems was inadequate attention to business processes, failure to improve processes before investing in information systems, investing in poorly planned and ineffective information systems, and outdated acquisition procedures that did not address the rapid evolution of information technology. (Deputy Assistant Secretary of Defense)

The Act mandates that federal agencies develop internal investment-control and performance-management processes to improve their acquisition, use, and management

of information systems. (U.S. GAO-00-179) The act established the positions of Chief Information Officer (CIO) for every major federal agency. The CIO became responsible for ensuring the provisions of the Clinger-Cohen Act are executed. Some of the responsibilities of the CIO were as follows: encourage incremental phased development instead of grand projects, ensure that the information system supports the core mission—as articulated in doctrine and policy—of the agency, determine whether other agencies or contractors have information systems with similar functionality as the system being developed, and perform cost-benefit analyses and risk assessments prior to embarking on developing an information system. Another key provision in the Act is the requirement to ensure that measures of performance (functional and non-functional) are used to gauge the effectiveness of information systems in meeting system requirements.

Furthermore, the Act requires software-acquisition personnel to answer three questions before initiating an IT project. The first two-part question is what are the functions that the system will perform, and is it consistent with the organization's mission? The second question is if we need to perform a particular function, can it be performed more efficiently and at a cheaper cost by the private sector? The third question is whether the function that is required can be reengineered or redesigned (i.e., are the processes it supports absolutely necessary)? All of these questions must be answered before an investment in new technology can go forward. (SecDef)

2. Difficulty Managing Technology

Despite the fact that the Clinger-Cohen Act requires the establishment of a process to identify, evaluate, and monitor risks and results from applying IT, the DoD is still having problems in both acquisition and management of information systems. (DoD IG D-2000-162) Since the Act was enacted, the DoD record on implementing its provisions has been disappointing. (DoD IG Semiannual Report to Congress) Some continuing problems with software acquisition have been attributed to the DoD's failure to adopt the provisions of the Act (DoD IG D-2000-162, DAWIA), and some was due to the DoD's current organizational structure and culture, which makes departmental oversight very difficult. (U.S. GAO OCG-99-4) Moreover, the DoD has not been able to

implement practices conformant to the Clinger-Cohen Act that ensure prudent investment in information technology. (U.S. GAO AIMD-00-282)

Notwithstanding the improvements that the DoD has made in the management of information technology, including establishing guidance to reflect best practices, and updating policies, the DoD continues to be plagued by problems in managing its portfolio of investments in information systems. (U.S. GAO AIMD-00-316) Unless the provisions of the Act are fully understood by program managers, fully supported by the chain of command, and enforced, it is unlikely that the Act will have the effect that Congress had hoped for.

For example, in 1994, the Under Secretary of Defense for Acquisition, Technology and Logistics mandated the use of "open systems," however, subsequent audits in 2000 revealed that fourteen of seventeen major weapon systems audited lacked open-system design objectives. Management either was not aware of the mandate, or they chose to ignore it. The DoD Inspector General (IG) identified management weakness along with poor analyses of requirements in twenty audits conducted between 1 April 2000 and 30 September 2000. (DoD IG Semiannual Report to Congress) The GAO has designated managing the investment in information technology as a major management challenge. (U.S. GAO HR-99-1, U.S. GAO HR-97-9, U.S. GAO 01-244, U.S. GAO OCG-99-4) The GAO identified a number of weaknesses in the DoD's management of its approximately 5,800 mission-critical or mission-essential information systems. (DoD IG D-2000-162)

Technology will not solve management problems. Program managers and senior leadership need to understand and improve business processes before applying technology. The GAO and the DoD IG have identified a number of systemic problems relating to the DoD's management of information systems. Of the programs audited, one of the most common problems was the lack of adequate documentation and validation of system requirements. DoD program managers do not always develop well-defined project purpose and scope, and realistic and measurable expectations. Audits also report the failure to perform risk assessments and develop appropriate risk mitigation strategies. Nine of the DoD IG audits identified inaccurate analyses of costs associated with the

system life-cycle. (DoD IG D-2000-162) An additional area of concern was the perceived weakness of the DoD in conducting information technology investment-selection and management-control processes. (U.S. GAO 01-244) The DoD's lack of centralized control over standards and architectures has also contributed to system failures. (DoD IG D-2001-121, U.S. GAO AIMD-00-282, U.S. GAO OCG-99-4) The DoD's inadequate software development, cost estimating, and system acquisition practices has greatly increased the risks associated with the information systems audited. (U.S. GAO AIMD-00-209R, U.S. GAO 01-244) The DoD has also shown significant computer security weaknesses in its programs. (U.S. GAO AIMD-00-295, U.S. GAO AIMD-00-188R)

Although the Clinger-Cohen Act established the position of CIO, the DoD needs to build an effective organization with the proper leadership. (DoD IG D-2000-162, U.S. GAO 01-244) Currently the DoD CIO and the CIOs in charge of the individual services do not control the budgets for IT. Individual programs procure their own IT systems and services to support their needs. As a result, CIOs often do not have the control or visibility they need to determine whether programs are complying with IT directives.

In its report to the Senate on adopted best practices for software development, the DoD stated that the responsibility for successful fielding of the software product was the responsibility of the contractor developing the system. However, in that report, the DoD could not state how it measures the success of a contractor's efforts. The DoD could also not state what requirements existed for maintenance or support. The DoD did list some generic metrics such as maintenance costs and number of problems reported, but it did not have clear guidelines as to what was acceptable performance for each of the quality metrics. (U.S. GAO AIMD-00-209R) Both the review and evaluation of performance metrics is essential in the acquisition of information systems (U.S. GAO T-AIMD/GGD-00-179), but requires knowledgeable information specialists working for the government to accomplish this task.

Shortcomings in information technology, contracting, and acquisition are attributable in part to human-capital issues. (U.S. GAO T-AIMD/GGD-00-179, U.S. GAO AIMD-00-282, U.S. GAO 01-244) The DoD IG semi-annual report to Congress

reported on the adverse consequences from cutting the acquisition workforce in half without a proportional decrease in workload. (DoD IG Semiannual Report to Congress) A shortage of personnel with the skill sets to manage IT intensive systems has also contributed to the lack of software quality. This is another reason that outsourcing has become more popular, although outsourcing efforts often require as much effort to manage as in-house efforts.

3. Shortage of Information-Technology Personnel

The DoD and industry have both been plagued by a shortage of workers with the IT skills necessary to support their organizations needs. Recruiting and retaining talented IT personnel is a problem for all organizations. In many cases personnel that are not familiar with IT have been forced into managing IT systems because there are not enough skilled personnel. This lack of IT knowledge has lead to many of the problems discussed in the previous section. It has also increased the reliance on contractor support and outsourcing.

In 1998 and again in 2000 Congress increased the quotas of H-1B visas in response to claims of a significant IT labor shortage from organizations such as the Information Technology Association of America (ITAA) and the U.S. Department of Commerce's Office of Technology Policy. (Matloff) In addition, The Department of Commerce projects a 1.3 million shortage in core IT workers by 2006. (Department of the Navy) In its 2002 study "Bouncing Back: Jobs, Skills and the Continuing Demand for IT Workers" the ITAA predicted that in 2002, of the projected demand for 1.15 million IT workers, 578,000 will go unfilled due to a lack of qualified workers.

Despite the amount of IT personnel that are currently unemployed, a recent study, and informal surveys have indicated that there still remains a shortage of IT personnel with the right skill sets necessary to help organizations achieve success in the complex, competitive IT market. (Griffith, Millard) The government has identified its largest IT skill gaps are in the areas of enterprise system integration and web-development. (U.S. GAO AIMD-00-282)

Part of the skill shortage is in the areas of IT program management. The are many program managers in the government's current workforce that lack the requisite skill sets

needed to administer the large, complex, software-intensive systems seen today. Many of the program managers are functional experts that have risen through the ranks to become program managers of major systems. There is no doubt that they understand the functional requirements of the system, but they do not have the training necessary to understand technical architectures, software documentation, software life-cycle management, or software engineering. In addition, with the current work load, it is difficult for program managers to keep abreast of the protocols, interface challenges, architecture constraints, or technological advancements associated with the move to distributed computing.

The DoD has shown that it is adept at utilizing risk management in systems engineering and the system-design process. However, it has not shown that same competency in software development. Experience has shown that the software component of major acquisitions is the source of most system risks. The software component is most frequently associated with late deliveries, cost escalation, and inefficient performance. (U.S. GAO AIMD-00-209R)

The GAO and DoD IG have acknowledged that the DoD does not have enough skilled information-technology workers to properly manage its information systems. The GAO expressed its concern that during the downsizing efforts in the DoD, more attention was paid to the reduction in numbers than managing the various skill sets of the workforce. (U.S. GAO 01-244) Thus, some people with necessary skills, such as information technology, were not been retained.

DoD, like industry, is having difficulty retaining skilled IT employees. The DoD civilian workforce is aging, and the GAO has identified retaining personnel with computer skills as one of the major managerial challenges for the DoD in the year 2001. (U.S. GAO 01-244) The mean age of the civil service workforce in the Department of the Navy (DoN) is forty-six, with nineteen years of service. Nearly fifty percent of the civilian workforce is approaching retirement. Of these civil service employees, one third of the civilian computer specialists will be eligible for retirement in 2003. (DON CIO)

The civilian workforce has declined about forty-three percent since 1989. (U.S. GAO 01-244) This downsizing in many cases has lead to the termination of the younger

employees. The policy of "bump and retreat" has forced many of the most junior personnel from the workforce. This policy, designed to protect senior workers, not only can lower morale among the existing entry-level workers, but it can discourage new accessions.

The DoD has difficulty in recruiting personnel to replace the civil service employees who retire. During good economic times, the salaries and benefits offered by the private sector for information-technology personnel outdistance those offered to government employees. The private sector offers from fifty to one hundred percent more for entry-level information-technology professionals than the government. (DON CIO) The advancement opportunities within DoD are limited due to downsizing, outsourcing, and the seniority of the existing staff. There is also a perception that junior information-technology professionals will be assigned to maintain legacy systems, rather than participating in the use of cutting-edge technology. As a result, there has been a decline in the number of young people who are pursuing careers in the civil service.

Most program managers control the functional aspects of the systems they manage well, but due to their lack of IT knowledge and the shortage of in-house IT support, they are forced to rely more on contractors to manage the software components of their systems, including maintenance. However, 0utsourcing IT functionality does not lessen a program manager's responsibility for managing that functionality. Program managers must still maintain control over their systems, they must be involved in the development and maintenance actions on their systems, ensure adherence to formal policies and procedures and provide contractual oversight.

4. Outsourcing

Outsourcing is the process of contracting with a service provider to perform a function or functions that used to be performed by the organizations own (in-house) staff. Outsourcing has been a business strategy for a number of years. Organizations are generally more comfortable assigning functionality to in-house staff as it gives them more flexibility, they do not need to contract for the services, in-house staff already understand the organization's policies and procedures, they have greater trust in their

own staff, and in many cases in-house staff is cheaper than contractors. However, in the IT industry outsourcing is becoming ever more appealing.

Many organizations have discovered that they do not have the necessary IT skills within their organization. Rather than hire IT specialists, or invest in training for their staff, they are considering outsourcing their IT work as a strategy. The emergence of ESPs have provided a source of IT specialists that can in many cases provide high quality service for lower prices than internal IT organizations can. IT outsourcing is gaining popularity and is increasing in volume worldwide. In many cases IT managers have little choice but to outsource as ESPs provide access to cutting edge technology and skilled staff, they share the project risk, and they allow organizations to concentrate on core competencies. (King, Goth, Greaver)

Numerous books and papers have addressed the topic of outsourcing IT . Research has addressed outsourcing of information systems from a number of perspectives. Some research has addressed the strategic implications of which information systems should be outsourced (Lacity, King, Beath, Nelson), others have written about the potential for offshore outsourcing efforts (Heeks, Smith, M., Kobitzsch), others have concentrated on the acquisition aspects of outsourcing (Farbey, Robert, Ripin), and some have addressed organizational risk (Duncan). Given manning shortfalls and a shortage of technical staff within the DoD, outsourcing IT services can increase the risk that the DoD's will not be able to provide proper oversight of the acquired service.

Currently program managers are increasingly forced to rely on contractors to provide technical guidance, because in-house expertise either does not exist, or it is overburdened supporting other programs. This has however, added another level of complexity to the management of information systems. Outsourcing efforts require additional discipline and management oversight that may not be necessary with in-house development and maintenance of information systems.

Outsourcing requires skill in software acquisition as well as project management. In many cases new processes must be created to manage the relationship between the organization and the outsourced contractor. Issues such as the level of access to

information, reporting chain, problem resolution procedures, reporting mechanisms, common software, and roles and responsibilities will have to be negotiated. In-house activities already have established operating procedures. Software acquisition also involves activities such as requirements determination, solicitation preparation, contractor and proposal evaluation, requirement change management, risk assessment, contract management and oversight, and contractor performance management. (SA-CMM)

F. PERFORMANCED-BASED SERVICE ACQUISITION (PBSA)

The Department of Defense has been shedding its internal development activities for a number of years. The DoD has moved from a producer of end-items to a consumer. Many of the services that were once performed by the military and DoD civilians are now being performed by commercial entities. Development activities such as SPAWAR and NAVAIR spend more of their effort managing outsourcing contracts than they do actually producing end-items.

As a result, acquisition of services and end-items has increased in importance due to the DoD's reliance on the commercial sector to meet its demands. To ensure that quality services or end items were being acquired, the government developed very detailed military specifications (Mil-Specs) and standards (Mil-Stds) that not only described their requirements, but it also described steps (processes and procedures) that the contractor needed to take to meet those requirements. Unfortunately the use of Mil-Specs and Mil-Stds did not necessarily result in a quality product. Eventually, the DoD stopped requiring most of the Mil-Specs and Mil-Stds because they were difficult to enforce, they were difficult to understand, they allowed the contractor little innovation or flexibility in meeting the requirements, they were not being used correctly, they were expensive, and the government was loosing the expertise to develop and enforce them.

After the DoD stopped utilizing Mil-Stds and Mil-Specs, their acquisition strategy concentrated on defining their requirements, and allowing the contractor to determine the method to best meet those requirements. The DoD strategy of creating requirements, passing them to a contractor to develop a product, then testing the final product did not result in improved quality. While this approach has a lot of advantages, including allowing contractors increased flexibility to derive solutions, it allows contractors to

utilize the best business procedures and latest technology, it increases innovation, and allows more contractors to compete for programs, it also has problems. One of the major problems is that the requirements have to be very explicit, they have to be unambiguous, quantifiable, and measurable; this is not always the case. Another problem with this approach is that the DoD advocates any responsibility for quality control until the test phase. This presents major problems if requirements were not met. This approach also does not foster good communication as requirements are "thrown over the wall" to the contractor, and discussions tend to be limited to better defining requirements and evaluations of the testing process and results. This approach lacks monitoring and quality control on the part of the government.

This strategy has been further refined into a new strategy called Performance-Based Service Acquisition (PBSA). Like the previous acquisition strategy, PBSA concentrates on defining service requirements in terms of performance objectives. PBSA does not dictate processes; instead it depends upon the contractor to determine the most effective and efficient means to deliver the requested service. A USD (AT&L) memorandum of 5 April, 2000 stated that at least 50 percent of service acquisition are to be performed under PBSA by 2005. (USD (AT&L))

While both strategies advocate early planning and spending the appropriate time to develop well-defined requirements, the difference in the strategies is that PBSA concentrates on stating measurable requirements, determining acceptable performance parameters, it requires a performance assessment plan to determine how contractor performance will be measured and assessed, and the PBSA also encourages the use of incentives (positive and/or negative reinforcements for meeting stated requirements). The PBSA also advocates a team approach in developing the requirements, as well as performing a risk analysis associated with the requirements and development proposals.

The PBSA strategy focuses on insight into the contractor's performance, not oversight. PBSA as opposed to the prior acquisition strategy encourages periodic assessment of contractor performance to promote quality control and enhance communication. This approach does not concern itself with the processes that the

contractor chooses to incorporate during development, but it does assess the deliverables resulting from the development process used.

The PBSA applies to the field of software acquisition as well. However, the PBSA strategy needs to be expanded to meet the unique needs associated with software acquisition. As the DoD has become more dependent on commercial sources to meet its software development needs, it needs to adopt a software acquisition strategy that emphasizes quality, not only in the end product, but also in project management, process control, and post-production support. This dissertation proposes the use of SLAs to achieve that end.

SLAs incorporate many of the elements of PBSA. In particular, SLAs support the performance assessment plan required by the PBSA approach. SLAs specify measurable performance thresholds, the methods by which the requirements will be measured, the periodicity of the monitoring, and incentives for meeting or failing to meet requirements. SLAs help to institutionalize many of the quality control measures that were lacking in prior acquisition approaches. SLAs focus on non-functional quality factors, while PBSA traditionally focuses on function requirements only. SLAs also encourages all stakeholders participate in the requirements engineering process.

While SLAs can be used to enhance PBSA, they can also be used to improve other software acquisition strategies in the commercial sector as well. As such, subsequent discussions in this dissertation will not specifically mention the PBSA approach. Instead, standard contracting terminology will be utilized. The remainder of this dissertation is intended to demonstrate how SLAs can be utilized to improve software quality.

G. SUMMARY

IT systems are the primary enabler to an organization's critical business processes. However, managing software-intensive information systems has been problematic for both DoD and industry. The difficulty recruiting and retaining skilled IT personnel, the rapid change of technology, and program manager's inexperience with IT has lead to software quality problems. Software quality has also suffered due to

organizations perceived need to rush software to market, poorly designed software, lack of programmer training, and dependence on testing to discover errors.

However, one of the primary reasons that many software-intensive information systems fail to meet expectations is due to the organization's lack of a quality control methodology. Program managers are not only responsible for defining the quality metrics that they need to ensure the success of their program, they must initiate the steps to ensure that quality is incorporated into the design, that quality is delivered, and that quality is maintained throughout its lifecycle.

II. SERVICE LEVEL AGREEMENTS

Service level agreements are becoming more common as organizations are relying on IT systems to provide their core business functionality. The increasing trend of outsourcing has also highlighted the need for a contractual mechanism, such as SLAs, which describes the services to be outsourced, but also holds the contractor accountable for their performance through penalties. This chapter will describe SLAs and provide some background on why they are becoming more popular. It will also illustrate a recommended format for the SLAs. The proposed format was a result of our extensive research and is designed specifically for IT system development, management, and lifecycle support. The chapter will conclude with a discussion on how SLAs can act as a framework to incorporate and integrate organizational and technical considerations.

A. DEFINITION

A SLA is a contractual agreement between a provider of services and a customer that defines a level of performance. (Aries, Strum, Factor, Surmacz) This agreement defines in measurable terms the service to be performed, the level of service that is acceptable, and the means to determine if the service is being provided at the agreed upon levels. SLAs define the quality of service, and how it is measured.

In general, there are two types of SLAs. The first is a contractual SLA and the second is an in-house SLA. The contractual SLA is used when dealing with third party providers or External Service Providers (ESPs) that are outside of the organization. In-house SLAs are used within an organization to describe the services the IT department provides to other departments. Both types of SLAs define the services offered in great detail, and are very explicit in stating customer expectations, however, contractual SLAs are more formal, and because of their legal implications, generally take more time to develop.

Contractual SLAs are used by organizations to specify their requirements and to protect their interests. Contractual SLAs usually have incentive or penalty clauses tied to the attainment of the service levels. These clauses provide the ‘teeth’ in the contract in

an effort to instill in the service provider a level of accountability. If organizations cannot receive the services that they specified in the contract, they will want some form of remediation. The remediation can be in the form on monetary penalties, or it may be an escalation of the issues to upper management for resolution. Some organizations try to avoid an adversarial relationship that penalties may cause by using incentives. An incentive clause may state that if an ESP meets all of the SLAs for a particular month, then an additional fifteen percent bonus will be added to the monthly payment. The goal of penalty or incentive clauses is to focus additional emphasis on meeting the quality thresholds or performance goals stated in the SLAs. In many cases if SLAs are not met, business processes are adversely impacted; it is not unreasonable that the ESP should share some of that risk.

SLAs explicitly define the services to be performed and the levels of service (this dissertation will also refer to levels of service as quality thresholds or performance levels) that an organization requires to support its underlying business processes. However, it is not uncommon to read service contracts that go to great lengths to define the services an organization requires, but neglect to include verbiage concerning the quality of those services. There are a number of reasons that quality thresholds are not specified in the contract, including time constraints and lack of clear requirements, but it is usually a result of the organization's lack of the technical expertise. If SLAs are not included in the contract, the customer can do little if the service levels do not meet their expectations. In many cases the customer has to tolerate the poor service until the contract expires, or the customer may be forced to renegotiate or terminate the contract.

When constructing a house, a contract may state that the upstairs shower must be functional before acceptance. However if the contract did not specify metrics by which to measure the term 'functional', the contractor could legally pipe the water into the shower with a ¼ inch pipe, or utilize a 10-gallon hot water heater, and still be in compliance with the contract. Fortunately, there are building codes that protect the consumer, but the same is not true in the IT arena. This is why SLAs are so important in IT acquisition.

The SLAs provide a common understanding on the services that will be performed, the levels of service are expected, how they will be measured, as well as define the responsibilities of both parties. Both parties must mutually agree upon contractual SLAs, or there will never be a contract. It is commonplace to negotiate on the services and the performance levels that are requested and ultimately agreed upon. A SLA should contain a definition of service requirement that is both achievable by the provider, and affordable by the customer. The customer and the ESP must also define a mutually acceptable set of indicators of the quality of service. (Sturm) It is important to note that SLAs can and should be modified throughout the lifecycle of a system as requirements change, technology improves, and efficiencies are gained.

The second type of SLA is an in-house SLA, which is used within an organization. This type of SLA provides the same type of information that a contractual SLA provides, but it is generally less formal. It is however, no less important. In-house SLAs specify the services and levels of performance that the internal IT department provides to other departments. These types of SLAs are becoming more common as they play an important role in quality control. The quality of services that the IT departments deliver are receiving more scrutiny as essential business processes are becoming more dependent upon the services delivered by the IT departments.

In some cases IT departments do not provide the services or the level of services that are needed by other departments, or they provide and charge for services that are not wanted. The in-house SLAs highlight the users needs, so the IT department can better align itself to providing those needs. (Hiles) The in-house SLAs ensure that departments get the level of service they need to support their requirements, the IT department can take the steps necessary to meet service levels that may exceed those currently being offered, and management can measure service against the agreed upon thresholds.

SLAs define an acceptable level of service that both parties agree to. Most program managers will demand 100 percent availability going into SLA development efforts. However, when they discover the costs associated with even 99.5 percent availability, they begin to relax their requirements. Program managers need to understand the levels of service associated with their current systems and the affect that

those levels have on their business processes, before they begin to develop SLAs for new services or systems. The in-house SLAs set a reasonable level of expectation that everyone, especially the end-users can understand.

In-house SLAs typically do not generally contain a lot of information on responsibilities or mediation procedures as those are usually covered elsewhere in the organization's policies. They also do not include penalty or incentive clauses. However, just because penalty clauses are not included does not mean that poor performance will not result in fiscal implications. In-house SLAs allow management to compare the costs of the IT department against the services they provide. If management is not satisfied with the performance of the IT department, these same SLAs can be used to determine if outsourcing may be a better option. Additionally, in-house SLAs provide a good business case for justifying positions, expenses, or needed capital investments. In-house SLAs are also an important part of an organization's quality control methodology.

B. BACKGROUND

SLAs originated from the dissatisfaction of users of IT services and the lack of objective measurements to assess service quality. (Hiles) Service level agreements are not a new concept, they have been around since the 1960s, however they are gaining more acceptance in both government and industry. There are a number of reasons that organizations are beginning to embrace SLAs. The main reason that SLAs have gained popularity is that there are now tools in the marketplace that provide the measurement capability to monitor SLA compliance. Another reason is that organizations have become increasingly dependent upon information technology (IT) to satisfy their business needs. As managers realize that their processes are tied to IT services, they are demanding more quality control over those services. One way to establish that control is through SLAs. The growing trend towards outsourcing IT functionality to ESPs has also encouraged SLAs as both a contract mechanism to define services, and as a marketing tool for the ESPs.

There has been a shift in industry from centralized funding of the IT department to handling the department as its own cost center. It is very difficult to allocate all of the

IT costs among the various business units. The direct costs associated with developing a specific project can be captured, as well as the costs associated with the software and hardware procured, the labor involved in the development and testing effort, and training can be captured. However indirect costs such as the costs associated with the entire network infrastructure, IT staff not directly associated with a project (e.g., firewall administrator), facilities, and help desk support are difficult to assign to an individual cost center. (Atre, Byron) The difficulty of assigning costs to individual departments resulted in many organizations centrally funding the IT department with little regard to the support provided to the other departments. However, as IT becomes more integrated in business processes, and IT costs continue to escalate, organizations are reassessing the way they perform IT accounting, resulting in reallocation of IT costs among the business units.

Organizations are increasingly under pressure to cut costs. Competition is fierce and all business units must justify expenditures in terms of benefits to the organization. IT departments must also justify their expenses. Unfortunately it is difficult to perform a cost-benefit study when expenditures cannot be tied to the specific business processes the funding is supporting. As a result, many IT departments have initiated charge back systems where business units are charged for the IT services that they require. (Chutchian-Ferranti, Ellett) Charge back is an effective mechanism for balancing the shape and quantity of the IT services with the requirements and resources of the business units. (ITIL p.64)

The main benefit of this type of IT accounting is that it provides management information on the costs of providing IT services that support the organization's business needs. This information is needed to enable IT and business managers to make decisions that ensure the IT service organization runs in a cost-effective manner. (ITIL)

Charge back systems focus a great deal of attention on the services that the IT department provides, and the quality of those services. Departments that pay for IT services want to quantify the levels of service, so they can determine whether the service is worth paying for. When individual business units are charged for IT services, an agreement must be developed between the business unit and the IT department that

outlines the services performed, the charge back mechanism utilized, and the level of services that the customer can expect. The agreement that is developed usually forms the core of the in-house SLA.

Even if a department is still funded centrally, organizations are demanding IT departments specify the services they provide, and the corresponding levels of service that other departments can expect. As IT systems become more pervasive in business, they are increasingly receiving scrutiny. The performance of the IT systems directly affects the business processes they support. Business managers need to know the level of performance they can expect from the IT systems. Utilizing SLAs, the levels of service are defined and the business impacts and financial repercussions of IT service levels can be identified and evaluated. SLAs have been a popular means of both defining the levels of service the IT system can provide, and providing remediation procedures if they fail to meet performance thresholds.

Monitoring tools consists of the software, hardware, agents, and databases used to collect and record information on the state of the underlying hardware, software, or infrastructure that provides the services specified in the SLA. In the past SLA performance thresholds were difficult to measure because good monitoring tools did not exist. Consequently, it was difficult for a customer to hold the service provider accountable for poor performance. As a result older SLAs were generally informal agreements that specified performance goals, but contractually they were very difficult to enforce.

Monitoring tools today are much more sophisticated. Products such as Hewlett-Packard's OpenView, Tivoli's Management Framework, and BMC's Patrol are pervasive in the IT industry. There are well over 800 vendors which market monitoring tools that measure performance. (Sturm) Unfortunately, few vendors can provide a complete monitoring solution. In many cases tools from multiple vendors may have to be utilized to ensure all services are adequately monitored.

Monitoring tools are bringing credibility to SLAs. Organizations are more willing to utilize SLAs when they realize that monitoring tools exist that can verify performance thresholds. Monitoring tools make SLAs more contractually binding; penalties or

incentives can be used more effectively to ensure that service levels are being adhered to. If a service cannot be adequately monitored to the satisfaction of both parties, it should not be included in a SLA as disputes will be difficult to resolve.

Organizations are outsourcing functionality for a number of reasons including cost reduction, taking advantage of commercial best practices, interoperability concerns with partners, utilizing technology that may not be otherwise available, and acquisition of expertise. (Loeb, Duncan, Greaver) Many organizations are struggling to keep up with the rapid technology change. Quality IT personnel are difficult to hire or retain, and it is hard to keep employees proficient in the latest technology.

Today's competitive pressures are forcing organizations to drive down costs and optimize on efficiency and effectiveness. If IT services such as infrastructure management, application development, application maintenance, and hosting activities can be outsourced to an organization that because of specialization or experience is more efficient and cost effective, then organizations must consider outsourcing as a strategic business tactic. It is also difficult to keep employees trained in the latest technology. (Feeny) Outsourcing IT functionality puts the risk and burden of managing a competent workforce on the service provider instead of the organization. This strategy also complements the fact that many organizations are focusing on their core competencies, or those IT services that offer the most strategic business advantage, and are outsourcing the remaining IT services needed by the organization.

The outsourcing decision generally revolves around a cost-benefit study, a review of business processes and strategies, a determination of the current levels of service (as opposed to those offered if the services are outsourced), reviewing core competencies, and an evaluation of opportunity costs. (Domberger, Norris) Issues such as costs to obtain the outsourced functionality or end product must be weighed against variables such as flexibility, complexity/uniqueness of the technology, business criticality, staffing skills, time criticality,risk, and organizational bias. (Nelson, King)

IT outsourcing has continued to experience significant growth. In 2000 the IT outsourcing market was worth over $100 billion. Outsourcing IT as a strategic business practice has gained credibility by its acceptance in many of the largest corporations.

(Kern, EDS) In addition, IT outsourcing is no longer just considering non-strategic services (e.g., those that do not affect business critical processes); businesses are now outsourcing strategic IT services. (Nelson, Duncan) As organizations begin to outsource business critical functionality to ESPs, SLAs become even more essential as they define the services to be provided, the performance levels associated with those services, responsibilities, and obligations of both parties. The lack of clearly defined requirements will ultimately lead to problems with the ESPs. There is much more to a good partnership than a contract, but the contract provides a foundation by which to develop the relationship.

It is important to make a subtle distinction between SLAs and requirements. SLAs are a subset of requirements and they are more contractually binding than requirements are. SLAs contain penalties and/or incentives if thresholds are or are not met. Other requirements do not have the same contractual rigor. In most contracts, the only recourse if a requirement is not met is to cancel the contract, or terminate any ongoing contractor support. Terminating a project is difficult, especially if the project is business critical. The difference between requirements and SLAs is the degree of recourse if a requirement is not met.

The major reason for the contractual nature of traditional SLAs has been the perceived need to penalize the ESP for nonconformance or failure to meet agreed upon threshold levels. The usefulness of penalties is subject to debate. Some believe that service rebates or penalties are difficult to enforce and are normally nominal in nature. The failure to hold ESPs accountable has reinforced the view that the contractual nature of SLAs restricts the scope and usefulness of such agreements without adding any significant value to the process. (Factor) Others feel that penalties focus management attention on the service quality and penalties provide a method to distribute risk to both parties.

Many ESPs have SLAs already developed for the services that they provide. Each level of service that they are willing to provide is priced out so organizations can select from a menu of services and service levels. However, it is not advisable to accept SLAs that are generated by the service provider. In most cases organizations should

generate their own SLAs, and negotiate to levels that satisfy both parties. The SLAs developed by the ESPs are generally very vague, usually do not provide access to monitoring tools or reports, rarely have penalty clauses associated with them, and ultimately are designed to favor the service provider. Additionally due to the vague nature of the SLAs, they are difficult to legally enforce. SLAs generated by the ESPs are usually marketing devises, designed to look appealing, but they almost always give the ESP a more favorable contractual position.

To date, the vast majority of SLAs have been written to cover services associated with the post-production support of an application (e.g., network services, help desk support, problems response). This dissertation proposes an original approach to software acquisition by utilizing SLAs throughout the lifecycle of a software-intensive system. Many of the advantages of utilizing SLAs in post-production support can be leveraged in requirements engineering, development, program management, and testing. This dissertation will demonstrate how SLAs can be used throughout a program's lifecycle to improve quality.

C. SLA FORMAT

SLAs serve as a mechanism to notify all parties of services that will be performed, performance expectations, responsibilities of all parties, penalties for non-performance, and SLA resolution procedures. SLAs also define the oversight and interaction between the program managers and the service provider.

Service level agreements have many formats depending upon how they are used. Internal SLAs between management and the IT department can be more informal because many of the procedural issues are stated elsewhere. SLAs involving ESPs need to be more formal.

There are numerous variations to the format of the SLAs, although most have a couple data elements in common. SLAs should describe the service to be provided in enough detail to ensure that both parties understand the requirement. The description of the service should be concise, understandable, and accurate. SLAs must also describe the performance thresholds for the services provided. Most SLAs will also contain data

elements describing the roles and responsibilities of both parties, penalties or rewards, escalation procedures, and assumptions. Good SLAs will also describe how the service level thresholds will be measured, which reports are required, data sources, and contract exceptions.

As was mentioned in the introduction of this dissertation, one of the original contributions of this dissertation is that it introduces a unique format for SLAs that combines some of the common elements found in SLAs with new elements that emphasize support for business processes, monitoring, conflict resolution, and identifying responsibilities. This section will outline the unique format of the SLAs that were used for the hosting services covered in appendix (A). The SLAs for hosting services added some additional data fields to provide clarity, ensure that the underlying business processes were being taken into consideration, and that there were people identified to validate the SLAs. The section that is indented is utilized for sub-services. For example if the service name is Help Desk Support, a sub-service category may be Customer Wait Time. If there is no sub-service, the indented section will be used with the main service category.

The following is the SLA template used in Appendix (A):

Service Name: This is the name of the service category that is being measured (e.g., help desk support).

Service Description: This is a detailed discussion of the service that is to be performed. The service should be as detailed as possible. In the government, the development team needs to be careful not to get to the level of detail where the government is telling the contractor how to perform the service.

Reason for Measuring: This section should provide the rational for this SLA. In this section the core, primary and secondary processes that are being supported by this specific SLA should be identified. This will help to justify the SLA, and it will help the program management team track which processes are tied to SLAs. This section is intended to ensure that the SLAs are linked to a strategic or tactical business concerns.

Time Frame: This is the time period during which measurements are taken (e.g., 24x7x365, or from 0700-1900 Monday through Friday)

Scope: This section defines where the services apply (e.g., this applies to the system software only). This section also provides amplifying information such as categorization of problem calls (e.g., priority 1 equates to an emergency), and information necessary to ensure all parties understand the areas that are covered by the SLA. The scope also details areas not covered by the SLAs.

Performance Category: This section names sub-services that must be measured to determine the over-all efficacy of the service. There can be numerous performance categories associated with one SLA. The following subsections are associated with every performance category:

Performance Metric: This section describes the metric that will be utilized to measure performance.

Threshold Levels: This section describes the various service levels that must be met. There can be multiple levels of service for each sub-service. In the NAVSUP hosting SLA, three service levels are used, corresponding to the essential, enhanced, and premier services as outlined in the SOW.

Formula: The formula describes how the metric(s) will be computed.

Assumptions: All assumptions that went into the development of the SLA should be stated in this section.

Contractor Responsibility: This section details the contractor's responsibilities in meeting the service level requirements.

Customer Responsibility: The program manager or the end-user's responsibilities are outlined in this section (e.g., a trouble call must be initiated before metrics covering the help desk can apply).

Frequency: This is the period of time over which measurements will be taken to determine SLA compliancy (e.g., monthly, quarterly). This usually equates to the periodicity of the reporting requirements.

Measurement Techniques: This describes the procedures that will be used to collect or verify whether the threshold levels have been met.

Reports Required: This section details the reports required from the service provider to verify actual performance against SLA thresholds. It also details the

periodicity requirements of the reports (e.g., Trouble Tickets – Monthly). In some cases, the person reviewing the SLAs has access to the report-generating tool, and can manipulate the reports as needed. An example is if the reviewer has online access to the trouble tickets, that individual can do daily, weekly or monthly reports, at whatever level of abstraction is needed. Details of the report contents, format, periodicity and distribution are detailed in the SOW or another document called the Contract Data Element Requirement (CDRL).

Person Responsible for Verification: This section details who will be reviewing the SLA measurements and determining compliancy. In the government, this person is usually the Contracting Technical Representative (CTR).

Escalation Procedures: This section describes actions to be taken when thresholds are exceeded, and who should be notified. For example if help desk response time is 15 minutes for a critical application, and 30 minutes have passed, who should be notified? This also includes situations where thresholds are violated on numerous occasions throughout the reporting period. Another use of this section is to describe the escalation procedures if the CTR and service provider cannot agree that a threshold violation has occurred.

Contractual Exceptions: This section describes any exceptions to the SLA. For example an emergency situation may require the service provider to violate a SLA threshold.

Penalties/Rewards: An SLA without penalties or rewards is nothing more than an agreement. SLAs must have a mechanism to enforce compliancy. This section describes what action will be taken if thresholds are violated, or if SLAs are met. It is important to identify minor and major thresholds to ensure that the service provider is taking action to correct the problems. If the service being performed is mission critical, it is helpful to have a termination clause to ensure thresholds are not violated multiple times.

D. SLAS AS A FRAMEWORK

This section will illustrate how SLAs can be used to bridge the gap between organizational factors (this term includes social, organizational and programmatic issues)

and the more traditional technical factors associated with software engineering. Early approaches to software engineering was based on the perception that modern scientific methods, with an emphasis on formalism, rationality, objectivity, and decomposition, could provide a solution to problems associated with software development. Software engineering was attempting to apply engineering approaches by applying objective standards to computer programs to test their correctness. Much of the early software engineering literature was associated with technical issues such as structured analysis and decomposition, modular structure, information hiding, reducing complexity, and process models intended to present a series of actions necessary to produce a quality product. (Ewusi-Mensah) However, this approach makes the assumption that real world problems can be isolated, rationalized, and solved utilizing technology. This assumption has not been correct to date, as the complexity of real world problems has evaded attempts at rationalization.

In real-world software development projects the final product must not only be technically sound, but it must meet stakeholder and organizational needs. Software projects are always embedded within an organizational context that includes organizational norms and culture, varying stakeholder perspectives, politics, economic considerations, as well as external business forces. Post-modernists believe that these organizational aspects must also be considered in the development of software, as a technically perfect software program is worthless if it does not meet the needs of the end-user. The social or organizational variables are often difficult to identify, and they are difficult to model. Organizational variables often present the largest problems in software development, and are the primary reason that software development fails (e.g., unrealistic project goals and objectives, project management and control problems, requirements churn, lack of executive support, and insufficient user involvement.) (Ewusi-Mensah)

A successful software development project depends upon many interacting variables including technical, economic, organizational, environmental, and managerial factors. Successful software projects take a holistic view of problem solving, incorporating technical considerations with the environment in which the problem is

framed. Andelfinger has developed a conceptual framework that helps understand the merging of technical and organizational factors in real world software development. His framework involves the concept of reflective practice where technical, social, organizational and economic perspectives are taken into consideration through problem solving and problem framing activities. (Andelfinger)

This dissertation also proposes a framework utilizing SLAs as a means to intertwine the organizational and technical factors associated with software development. Project success depends upon three main factors: the design must satisfy user needs, there must be collaboration between users and designers throughout the development process, and finally there must be constant communication between designers and users to ensure prompt resolution of conflicts and misunderstandings. (Ewusi-Mensah)

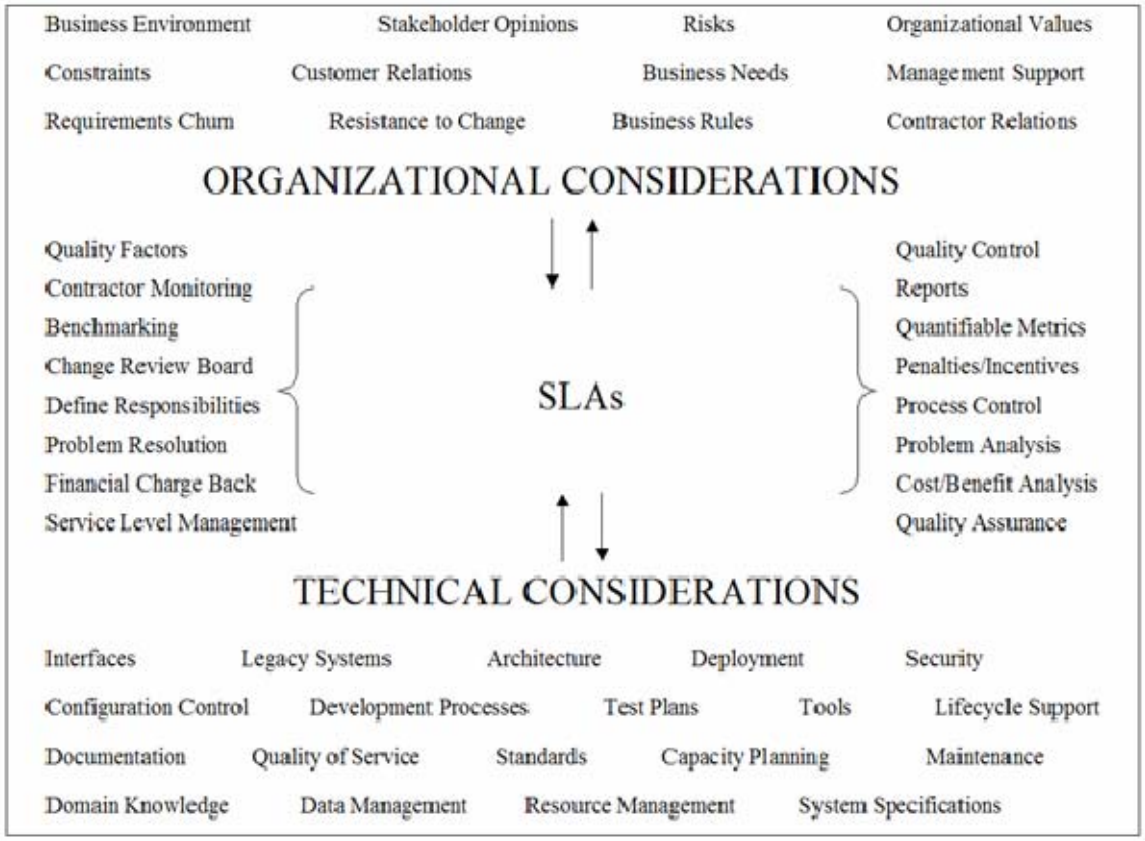


FIGURE 1. SLA FRAMEWORK

The SLA development efforts and subsequent quality control efforts associated with SLAs not only produces meaningful and measurable requirements, but the monitoring efforts encourage constant communication. Figure 1 provides a framework that illustrates how elements of SLAs and the activities associated with managing the SLAs help the program manager factor in organizational considerations and technical considerations in the problem solving process.

To achieve a successful project, the program manager must understand how organizational factors can influence technical considerations and visa versa. While this framework will not be discussed in further detail, it was presented at this point to provide a foundation. When reading subsequent chapters this framework may be helpful to see how SLAs can help the program manager develop a quality solution to the problem proposed, while accounting for technical and environmental factors. The SLA development process discussed in the next chapter will illustrate how technical and organizational factors must be taken into account in the requirements engineering phase of development.

E. SUMMARY

SLAs were developed as a means to reinforce contractual provision to increase the probability that the services provided by a contractor or the IT department meets the quality requirements necessary to support the underlying business process. The SLAs describe the services to be provided, the levels of service that must be attained, quantifiable metrics to validate compliance, responsibilities of both parties, and penalties or incentives associated with meeting or failing to meet service levels. Service level agreements improve quality by identifying quantifiable quality requirements that are incorporated into the requirements engineering process, and ensuring the test strategy evaluates the implementation of those quality factors from design to deployment.

SLAs are gaining in popularity as outsourcing is becoming more common. The owners of business processes are tying to gain more control over the IT services that support their business. Financial personnel are looking at SLAs as a means to allocate service costs to the appropriate cost centers. SLAs are also becoming more popular

because there are now commercial tools that are capable of performing the monitoring functions required by SLAs.

The format of the SLAs presented in this dissertation are unique in that they not only help to tie the quality requirements back to the underlying business processes, they also help to establish quality controls necessary to monitor contractor performance. The SLAs elements incorporate many of the organizational and technical considerations that affect the project. As such, the SLAs provide a framework for generating the communication and oversight necessary to identify and monitor technical and organizational risks and challenges.

III. APPLYING SLAS

This chapter proposes an 8-step process to develop SLAs that is applicable to most projects. This process helps to identify constraints that may make applying SLAs difficult, it determines those quality factors that are necessary to support the system, and it prepares the development team for the negotiation phase. This chapter will also discuss common traits found in successful SLAs. The last section is a detailed case study that illustrates the approach utilized to develop the SLAs contained in appendix (A).

A. DEVELOPMENT

There are numerous methodologies for developing SLAs. The approaches to development vary due to organizational culture, the type of SLA, the skill sets of the personnel involved, and the criticality of the process affected by the SLA. However, there exist some common steps that need to be addressed that span most SLA development efforts.

1. Define the Problem

Before SLAs are developed, management and the program management team must determine whether they should be used at all. While it is intuitive that SLAs should be used for outsourcing to ESPs, resource constraints, lack of management support, and lack of the appropriate skill sets may make the effort of developing the SLAs wasteful. The same is true for in-house SLAs, they may cause more problems than they are solving.

Charles F. Kettering stated that a problem well stated is a problem half solved. The first step in developing SLAs is to define the problem that the SLA is supposed to solve. When the SLAs involve ESPs, then the problem solved by the SLAs is how an organization can ensure that the services provided by a third party meet requirements. SLAs help solve the problem by explicitly defining the services, the quality of the service required, responsibilities of the parties, and methods to measure service levels.

When dealing with in-house SLAs, the problems become more difficult to define. SLAs used within an organization should be solving problems such as explicitly stating

the services required by various departments, producing measurable quantifiable data to support a level of service, and improving communications by explicitly stating service levels. SLAs can also be used by the budgeting personnel to tie the costs of IT services to the business processes those services support. This makes cost/benefit analysis much easier. SLAs can also help the IT department justify infrastructure or capital improvement expenditures by linking IT service costs to the underlying business process the services support. Unfortunately, in-house SLAs can also be used for political reasons.

In-house SLAs should be invoked as part of an organization's quality management initiative or program. In-house SLAs will not necessarily make a poor performing IT department better, but it will identify problem areas so management can address those issues. Some IT departments do not like SLAs because they feel that other departments use them as a hammer every time an SLA is not met. In situations where internal power struggles are common and the environment is highly competitive, SLAs may put the IT director at a disadvantage by tying that individual's performance to quantifiable metrics, while the other directors are not. Additionally, in some cases the IT department may not have input into the SLA, they may be dictated from upper management.

Another important issue to evaluate is whether upper management will support the SLAs. Service-level management (SLM) in the context of SLAs deals with the generation and oversight of the SLA contract, and ensures that the agreed upon services are delivered within acceptable thresholds. SLM must have management support and resources to succeed. The world's best SLAs will fail if there is not someone or a group of people that are responsible for monitoring, revising, and enforcing the SLAs. SLM generally requires additional personnel to provide the oversight necessary. If management is not willing to hire additional personnel or reassign personnel within the organization, the SLAs will not have the impact needed to ensure quality.

SLM also requires personnel with the skill sets necessary to understand the technical issues associated with the SLAs. With in-house SLAs, these personnel should not be solely from the IT department, as that is tantamount to the fox guarding the hen house. It is often difficult for organizations to find personnel with the skills necessary to

contribute to SLM that are outside of the IT department. Management must be willing to contract or hire the personnel with the skill sets necessary to provide the proper level of SLM.

It is important that the end users and business process owners understand the level of services that are necessary to support their business processes. If end users or the process owners are not willing to devote the time necessary to develop the SLAs, or if management is not willing to bring all stakeholders into the SLA development process, then little benefit will be gained from developing SLA that may not support the underlying business processes.

Knowledge of the business processes supported by the IT system is critical in developing the SLAs. Developing SLAs for services that do not have a direct impact on the business process may not be worth the effort. In some situations, external forces have more influence on a business process than the IT services that would be covered in the SLA. Resource constraints, fiscal constraints, market forces, and other variables can render even the best SLAs meaningless. If SLAs cannot improve the quality or performance of the supported business process, then the SLAs should not be pursued.

Upper management must be willing to take action if SLAs are not adhered to. With in-house SLAs, upper management must be willing to take action if the IT department continually fails to meet SLAs. This may be an indication that the SLAs are unrealistic, but it could also be that the IT department is not allocating the assets or attention to solve the problem. If the problem is the latter, management must take action; otherwise the SLA will have no value, and the end users will quickly become disillusioned. In the case of contractual SLAs, management must be willing to enforce penalties, or withhold incentives. In some cases, contracting personnel within the organization are not willing to perform the work necessary to monitor contractor performance, document problems, and take the actions necessary to ensure requirements are met. The team must understand the environment in which they are working before embarking on the efforts to develop the SLAs. If they are not going to receive the support they need, SLA development should not be started.

The team developing the SLAs must weigh the costs and time of developing the SLAs against the intended benefits. SLAs are essential when dealing with ESPs, but management must devote the proper resources to perform contractual oversight. Without monitoring and enforcement, contractual SLAs become nothing more than goals. In-house SLAs should only be attempted with managerial concurrence, and the agreement of both the IT department and the recipients of the service. Without agreement, the SLAs can cause more problems than they solve.

2. Develop a Team

Once the decision to proceed with SLA development is made, the next step is to create a team to develop or review a proposed SLA. This team should consist of all stakeholders. At a minimum representatives from the IT department and the recipient of the services need to be represented. The recipients can be individual programs or entire departments. Representatives from ESPs do not need to be included in discussions at this stage, although they can be. The team members should be able to contribute to the development of the SLA. From the end users perspective, their members should understand the business processes, application functionality, and the services needed to support their requirements. The IT department needs personnel that understand the technical aspects of the services offered, the quality levels they are capable of providing, and monitoring tools necessary to ensure delivery.

The team structure will vary with every organization, but there are a couple important elements that will help the development process. The team leads from the IT department and the user community should be on the same level, and should have decision-making authority. There should be a charter outlining the membership, responsibility of the team, leadership, structure, chain of command, and deliverable. The team should have a specific amount of time in which to deliver the SLAs, or review a proposed SLA. Representation on the team is needed from each stakeholder group, but the team should be as small as possible. A representative team would consist of membership from the IT department, program manager's organization, management, the

business process owner and end users (personnel inputting information or products into the business, or recipients of the output of the process). In a medium organization four to ten people is typical. (Sturm)

3. Service-Level Management

Service level management (SLM) is the disciplined process of ensuring that adequate levels of service are delivered to all IT users. (Sturm) SLM normally refers to the procedures and methodology that the IT department or an ESP utilizes to ensure that the services they provide meet specified service levels. In the context of developing SLAs, service-level management refers to the process of managing the SLA contract. SLM involves validating the levels of service against the quality thresholds outlined in the SLA, coordinating the change management process, evaluating the performance reports, and managing the business relationship with contractors and process owners.

The development team needs to determine the SLM functions that need to be performed, and then they need to scope those functions to determine the resources to allocate to ensuring that tasks are successfully executed. Then the development team needs to get management support to ensure that there are people assigned to perform those functions. SLM is resource intensive. If the development team becomes resource constrained, they may have to scale back the number of SLAs, modify their oversight roles, or decide not to proceed with developing SLAs.

As part of the SLA development process, the development team must determine how to verify whether service levels have been met. Depending upon the services provided, there are many ways to validate performance. In some cases there are automated tools that will assist in the verification process. In other situations, someone may have to review the raw data in server logs to determine compliance. Another common verification technique is to audit the contractor's processes for compliance. If customer satisfaction is a part of the SLA, someone needs to be responsible for administering the survey and compiling the data. One of the SLAs for backup tape accuracy requires that the contracting technical representative (CTR) physically audit the

backup tapes to ensure that they are properly documented, and that they are not corrupted. Depending upon the scale of the contract, multiple people can be involved in monitoring and verifying service levels.

The person responsible for managing the contract should also be identified. This person will play an important role in managing the business relationship as well as being a key member of the change review board. Any changes that impact the service levels, or computing resources can involve additional contractual modifications as well as funding. If contract modifications are necessary, the program manager will work with the contracting official to develop and negotiate the modification. The contract manager is also responsible for mediating any disputes between the customer and the service provider. Any escalation procedures should involve this individual. In the case of the hosting SLAs, the person identified to deal with escalation procedures is the Contracting Officer Representative (COR).

The SLA development team along with the program manager should determine the representatives needed at the change review board. At the very least the program management staff needs representation, the contract manager, the fiscal manager, the person or people responsible for monitoring the service levels, the user community and technical representatives from the IT department should be represented along with the service provider. Depending upon the requirements volatility associated with the program, the meeting could be held weekly. Additionally, the program manager's staff and the IT department personnel need to determine before the meeting the affect that changing requirements are going to have on the SLAs. For example if the application is going to be used by another command, and the concurrent user count is going to double, then the service provider will have a good case for requesting additional funds to purchase hardware for load balancing. In some cases the change review boards can involve discussions on the need for additional services or the need to modify existing service levels. The man-hours associated with these meeting, and the preparation for the meeting needs to be considered.

SLAs require considerable time and resources from the program management staff. If service level agreements have not been used in the past, the program

management group responsible for the development of an application, or the fielding of the application is going to have to devote additional time to developing, reviewing, or modifying template (already existing) service levels. The program management staff will have to participate in the development of the SLAs. They will also have to review the SLA reports, attend the change review boards, attend SLA review meetings, and spend time managing the relationship with the service provider. As service level reports are distributed to the user community and upper management, the program management staff will be forced to be more involved in managing the performance of the service provider. The program manager is expected to take action if performance does not meet service levels. The process of managing service provider performance will be much more labor intensive under SLAs than before. The program managers and the development team need to make sure that there are proper assets in place to handle this additional workload.

IT accounting personnel will also be tasked with additional work when SLAs are deployed. Procedures should be developed for how to handle the penalty or incentive provisions in the contract. They need to determine whether funds are budgeted up front anticipating incentives, or whether additional funds will have allocated if incentives are warranted. If requirements change drives new SLA services, or capacity, they need to determine whether there are there enough funds to cover the costs. The IT accounting personnel will have to work closely with the program manager and the COR to ensure that contract modification will not exceed the budget, and if they do, they will assist in preparing the justification for the financial review.

Personnel involved in SLM need to also constantly review the service levels against the underlying business process. They need to determine if the service levels are in fact supporting the business processes, or whether they need to be modified. Additionally, it is possible that some services though to be essential to the performance of the business process are in fact not needed. It is also possible that some service will have to be added to the SLA because they were not though of previously, or because additional requirements were added to the application.

SLM is the process that an organization utilizes to ensure that the contractor adheres to the requirements in the SLA. Poor SLM will undermine the efforts of establishing the SLAs in the first place. When developing the SLAs, the development team needs to not only identify manpower shortfalls, but they need to brief management and the program manager of the roles and responsibilities that they are expected to perform. The development team must also assess whether they have personnel with the skill sets necessary to verify service performance. If management or the program managers are not willing to allocate the time or resources, then the development team must determine whether to proceed with developing the SLAs. If the service levels are not monitored and verified by the customer, then they will quickly loose their effectiveness. The trust between the end users and the program manager will quickly erode. Users will become frustrated when service quality is poor, and the service provider will quickly determine that they will not be held to the threshold standards.

4. Review Current Services

SLAs can be utilized for the development of new systems, maintenance of existing legacy systems, or for post-production support. They can also be used for outsourcing services that were previously performed in-house. Before the SLAs are developed, it is important that the team has a foundation understanding of services and service levels that are currently being used within the organization. Once that foundation is built, services and service levels can be evaluated and applied to the new system, outsourcing project, maintenance action, or in-house project under consideration.

The development team needs to understand the underlying business processes that the IT system must support or enable. The team needs to not only understand the main process being supported, but it must also evaluate the numerous interlinked, feeder, and cascading processes it supports, or is being supported by. When evaluating processes it is useful to divide the processes into the core business process, primary supporting processes, and secondary supporting processes. The core business describes the end-to-end activities involved in supplying a deliverable or a service. The primary supporting processes are those sub-activities, organized in a logical sequence, that make up the core business process. The secondary processes are those activities that support (directly and

indirectly) the primary processes. (Tricker) It is difficult to control quality unless the quality objectives of the core, primary and secondary processes are defined.

When developing the SLAs the team must determine the organization's key business processes and determine the types and levels of service that are needed to support those processes. It is difficult to develop SLAs without first knowing what services are being provided, and at what level. The team should develop a list of all of the services currently supporting the primary and secondary processes, and then try to define quality levels associated with each of the services. The list of services should be as extensive as possible. If the team is reviewing services that are currently being offered by an ESP, a review of the existing contract, interviews with end users and ESP personnel, and a review of any required reports will be helpful. Interviews with the end users are especially important because many of the users may not be aware of the contract, and they may not be receiving services that they should be.

If the SLA is to be used internally, the IT department should list all of the services that they provide (relating to supporting the business processes). This is their opportunity to show all of the work that goes into providing their current services. There are many functions that must be performed that end users may not be aware of such as 24 X 7 physical security, monitoring of hardware and software, application testing, configuration management, or tuning the server to optimize application performance.

The SLA development team must also interview end users to determine what services they are in fact receiving. There may be differences between what the IT department claims they are providing and services the end users say they are receiving. The SLA development team must determine reality by observation and reviewing reports, trouble tickets, logs, and monitoring tools.

Once a list of services has been developed, the next step is to define the quality of the service. Each service should have a quantitative measurement of quality. However, it is not uncommon to discover that an organization does not have defined levels of service. If service levels have not been previously defined, the SLA development team will have to determine them. Interviews, observation, or benchmark testing will have to be performed to determine the level of service that is currently provided.

Benchmark testing is typically used in measuring performance based services such as application response time, network bandwidth utilization, or processor capabilities. However, benchmark tests can be utilized to measure service levels such as file retrieval, disaster recovery, or trouble ticket resolution times. Benchmark testing not only helps to quantify the level of service, but it also helps verify that defined levels of service are actually being met.

If the SLA development team is not comfortable relying upon the IT department to perform the benchmark tests, they may find it advantageous to contract with a third party to perform the benchmark testing. In some cases a third party may be necessary because the current IT department is not trained on the necessary monitoring tools, they do not have the background to develop a benchmark testing plan, or because the licensing costs of the monitoring tools are prohibitive. A third party would also provide impartial results that may make lessen conflict between the IT provider and the end users.

In some cases it is very difficult to assign quantitative values to the services that are provided. In some cases the services will have to be rolled into a higher service. For example the service 'tuning a server' may have to be rolled into the service 'availability' for that server. Conversely services such as 'security' may need to be broken into smaller services such as 'data integrity'.

5. Determine Requirements

Once the SLA development team has determined the services that are being provided, and at what level, they must determine if those services and service levels are appropriate for the business processes they support or are intended to support. Additionally, the team must determine if additional services are required, or if some current services can be deleted. New services must be defined, quantified, and assigned a level of quality that meets every stakeholder's needs.

IT managers need to understand their customer's requirements in order to provide the services necessary to meet those requirements. However, it is not uncommon for IT managers to make assumptions about customer requirements. IT managers often make IT investments based on customer's past requirements, customer's perceived future requirements, or they plan for improvements to the IT infrastructure to meet their own

needs. (Briones) Software cannot function in isolation from the system in which it is embedded, thus a systems level view must be used when performing requirements analysis. (Neseibeh) A purely technical approach without regard to the underlying business processes that IT supports will not satisfy the end user's needs. The end users, management and the IT department must be involved in the requirements analysis process to ensure that the services needed are identified, that they support the current and future business processes, and that the IT department can provide those services. The team approach to developing SLAs is essential in producing a product that is workable for all stakeholders.

If the SLA concerns the development of a new system, it is important for the team to understand the core, primary and secondary business processes that the IT systems (hardware, software, and infrastructure) are supporting. Part of this analysis is to gather information on the business processes that the IT system is enabling. The team can start by asking some simple questions. Is the process data query, data input, e-commerce, real-time collaboration, report generation, information sharing, or data warehousing? How does this process tie into the organization's business strategy? Is this a dynamic process or a relatively stable process? Is the information used by the process internal or external to the organization? If the information is external, what is the source, who controls it, and how is the information extracted? Is the data sensitive? How does this process tie into the overall IT architecture? Does the process have to interface with any other processes? How do they interface? In two years, how might this process change? Do people outside of the organization (e.g., partners, suppliers, customers) need access to the data? How old is the technology supporting this process? Are there manual processes in addition to those being automated?

The team must then determine how the application is or is intended to be utilized. Interviews will help determine batch processing times, the amount of response time that is acceptable to users and management, the hours that the end users actually use the application, location of the users, methods for accessing the application (e.g., intranet,

internet, remote dial-in), and timeframes for required reports or queries. It is also helpful to understand how downtime or reduced capabilities will affect the end user's ability to perform their tasks.

The team should also analyze the business criticality of the system from the end user and management's perspective. The financial implications of downtime should be determined so an accurate cost/benefit analysis can be performed. Implications of downtime can include not only lost sales and clientele, but also frustration and lost productivity by the organization's staff. In some cases, especially those in the military, the implications of downtime could cost lives. Highly critical business systems should also be viewed in terms of information assurance to protect both the data and the system itself from external and internal threats.

The business criticality of the business process gives the team a good indication of the types of services needed by the application, as well as how much funding the organization is willing to invest in those services. Applications considered business critical will be capable of justifying a larger budget, and consequently will be able to request more services at higher quality levels. If the application is being phased out for another application that works more effectively with partners or customers, then the services needed may be less that those needed by the replacement application.

Administrative requirements also need to be addressed in the SLAs. Program managers want the ability to quickly monitor the contractor and IT system performance to ensure they are meeting requirements, so the SLAs must address the reports that are required from the service provider. Reports are the vehicle to demonstrate whether actual performance met, that which was required. The team needs to determine who will be reviewing the reports. The reports (generated by the contractor, CTR, or through access to monitoring tools) will need to reflect the proper layer of abstraction to meet the manager's needs. Management may not understand the technical details of the reports, so they may need summary reports, whereas the personnel verifying the SLAs may need very granular data. The team will need to determine the content of the reports, their frequency, their distribution, the source of the reports, who prepares the reports, the report format, how the report relates to the measurement of the service, and how the

report can be verified. Any current reports can provide a baseline to determine the level of detail required, an acceptable periodicity, and management's comfort with the formats.

The development team must not only determine the services and service levels associated with product quality, but they must also incorporate any process, project, or post-production quality requirements into the SLAs. Reviewing SLAs that other companies have written for similar projects (template SLAs), or reviewing the contractor's SLAs can help identify services that the development team may not have considered.

Template SLAs can significantly reduce the time spend developing SLAs as they already contain definitions of services, they have quality thresholds that at least one organization found acceptable (hopefully, industry standards can be developed for certain SLAs), they contain the methodology to measure the service, and they explicitly state the assumptions that were used when developing the SLAs. Template SLAs provide a good framework to use. The development team can then modify the template SLA to incorporate the organization's requirements.

Benchmark testing produces information on the levels of service that are currently being provided. The requirements analysis further defines those services to determine if the services are needed, and if they are needed, whether the levels of service are adequate to support the application. Requirements analysis also determines if new services are needed and defines their associated level of service. Once requirements are defined, the team needs to be prepared to negotiate on the service levels. Realistic maximum and minimum thresholds should be developed for each service. Depending upon the costs associated with the maximum threshold, the team may decide to reduce the threshold level to at or near the minimum.

In most organizations, all costs must be justified, and as such, the team must be prepared to justify all of the services and their corresponding levels of service. The justification should be directly related to the primary or secondary process supporting the core business process. The team should be able to explain the business impact of the various levels of service. If resources are limited, the team should be prepared to

compromise requirements, so they should be prioritized. Before negotiations begin with either the IT department or an ESP, a draft SLA should be prepared.

6. SLA Preparation

Once requirements have been defined, and service thresholds have been established, the team can start to prepare the SLA. In this stage, the SLA format must be decided determined, then populated with all of the required information. This can be a difficult task, as the team will have to determine meaningful, measurable, and quantifiable metrics to measure the services needed. They will also have to define the scope of the contract, the services that must be performed, the service level thresholds, and all other required fields.

Part of the development process is to determine the format of the SLA. A recommended format will be presented later in this chapter. This same format was used in the SLA for post-production support in appendix (A). However, there are numerous formats that can be utilized depending upon the services requested, whether the SLA is in-house or contractual, and the needs of the organization.

The SLA development team needs to determine how the service required will be measured to ensure that the service levels are being adhered to. The customer should never rely upon the service provider to determine whether the SLAs have been met. The team must determine if monitoring tools, logs, software agents, or monitoring software packages are available to provide the information necessary to verify service levels. It may be necessary to perform audits or run benchmark tests to determine performance. The team should also determine if there are personnel in the organization (outside of the IT department if in-house SLAs are used) with the technical expertise to perform the audits.

If the team is not experienced in developing SLAs, or if they lack the technical expertise necessary to determine how SLAs should be enforced, they should hire consultants that work with SLAs or outsourcing contracting. Consultants can assist in determining the types of performance reports that should be generated by the service provider, and the means to audit those reports.

Services that cannot be measured or verified should not be included in the SLA. Those types of services should be listed in the SOW. In some cases the determination of performance is subjective, and it is difficult to get an objective measurement that both parties agree to. In some cases a survey can be used to determine an overall subjective measurement regarding attributes such as customer satisfaction. So long as the sample size is agreed upon, statistics can generate mean scores, which can be used in a SLA. Proxy attributes may be used to measure the performance.

Proxy attributes attempt to assign objective attributes to a subjective objective. A proxy attribute does not directly measure an objective, but can be used to describe the degree to which an objective has been met. It indirectly measures an objective. Rather than explain the Bayesian theory and probability distributions used, an example illustrates the concept better.

The overall concept of security is a subjective one. Many of today's IT systems are comprised of distributed, heterogeneous systems that pull information from multiple sources. There is not one simple measurement to determine if a system is secure or not. There are many objective indicators that can indicate a degree of confidence in a systems ability to withstand an attack. Attributes such as all servers are set up in accordance with the National Security Agency (NSA) approved configurations, the firewall is configured in accordance with the Navy Firewall Policy, adherence to Common Criteria guidelines, and intrusion detection software is deployed within the system, generates a measure of confidence in the security of the system. However, that confidence is still subjective.

None of those attributes directly measures security, but they can provide objective values that can be used to calculate a level of confidence in the security. It is ultimately up to the team and the service provider to determine if the proxy attributes can adequately be used to measure security. This means that the team and service provider must be able to understand the implication and extent that the proxy attributes relate to security. The goal is to provide as much objective information as possible so that a decision regarding compliance with a service can be justified.

The next step in developing the SLA is to determine who will be responsible for ensuring SLA thresholds are being met. That individual or team of individuals must have

the authority and resources necessary to provide the oversight necessary to audit performance and enforce noncompliance. Managing service levels can be a time consuming effort and cannot usually be assigned to the program manager of an IT system. In some organizations, a quality assurance department is responsible for SLM. The development team and the program manager must review the level of work necessary to perform the intended SLM functions when assigning the individual or individuals necessary to monitor SLAs. In many cases multiple people will be employed in the SLM effort

The SLA development team also needs to determine the scope of the SLA. The boundaries of the agreement need to be defined. This seems straight forward, but in some cases the service provider may have not control all aspects of an IT system's performance. A good example is where a service provider is being tasked to host an application in its server environment. The SLA specifies a threshold of a 2 second response time for a specific query in a client-server architecture. In this case the service provider has no control over the client PC, the client network to the Internet Service Provider (ISP), or from the ISP to the service provider's firewall. In this case, the scope should be defined to the service area that the service provider actually has control over.

Once the team has developed the SLAs, they are almost ready for the negotiation phase. The last step is to present the draft SLAs to the organizations attorneys. The attorneys will review the SLAs as they would any contract between the organization and a third party provider. They will undoubtedly modify the SLAs to add clarity and ensure there is verbiage to protect the organization if the services specified in the SLA are not delivered at the thresholds specified.

When an organization wants a contractor to propose a bid for the services that they want accomplished they prepare a request for proposal (RFP). In this dissertation the RFP sent to the service provider will include the Statement of Work (SOW) and the SLAs in separate sections of the RFP. It is important that the development team is familiar with the SOW. The SOW can define services that will be performed, but the SOW really concentrates on the functional requirements of the system. SLAs concentrate

more on the non-functional, quality requirements of the system. The SLAs should support the SOW, not conflict with it.

7. Negotiation

The SLAs must be agreed upon by both parties in order to be successful. SLAs that give undue advantage to either the organization or the service provider will cause problems. As service levels are not achieved, or expectations are not met, disputes and finger pointing ultimately occur. SLAs should not be viewed so much as a contractual mechanism to force the service provider into compliancy, but as a contract that defines expectations for both parties.

The contracting officials are generally responsible for leading the contract negotiations. SLAs are contracts, and as such, members of the contracting branch or department should be part of the development effort, or they should at the very least review the draft before negotiation processes begin. The program management team and the contracting official needs to determine if the process owners, IT personnel, contracting personnel, or management will be involved with the negotiations. Although the SLAs and SOW will probably be negotiated as a package, it is recommended that if they are negotiated separately, whoever negotiates the SLAs is also the same person or group that negotiates the SOW. This provides consistency and helps to ensure that the SLAs and SOW do not conflict. (Sopko)

Once the SLAs are drafted, they are incorporated into a Request for Proposal (RFP) along with the SOW. In government contracting section H is where the SLAs are placed. Section H provides additional guidance to the SOW. The contractors respond to the RFP and the SOW with a proposal that lists the services that they will provide along with the technical specifications on how they will achieve those services. The contractor must also respond to the SLAs. The contractors must not only determine whether they are capable of providing the services, but they must also be capable of performing to the service levels defined in the SLAs. In-house SLAs are usually presented to the head of the IT department for consideration.

It is important that both parties understand the terminology and technology that is associated with the SLA. Both parties need to understand and agree upon the verbiage in

the SLA. If there are areas that need clarification, then mutually determined modifications will have to be made. This may entail several meetings, especially when attorneys are involved.

The service providers must evaluate the SLAs for software and hardware requirement, staffing needs in terms of skill sets and effort to accomplish requirements, infrastructure needs, and managerial oversight needed. Once the service providers understands the hardware, software and resources needed to satisfy the service thresholds outlined in the SLAs, they can start to determine the costs associated with providing those services. They can also start to estimate the time frames associated with software development or software maintenance projects.

The service providers will also look at the deliverables and the responsibilities of both parties as defined in the SLA. Every section of the SLA is subject to negotiation, as this is a contractual document that is legally binding. Any areas that are subject to interpretation should be defined as much as possible to ensure both sides understand the services to be delivered. Attorneys from the service provider will also review the SLAs and they will play a role in the negotiation process.

It is extremely important that the development team have good information from their benchmark studies. The team should know the service levels that are currently being received and to the maximum extent possible, they should know or be able to estimate the costs associated with providing those services. If that information is not known, then the team is entering the negotiation process in the blind. The team will not be able to determine whether the services requested exceed requirements, nor will they be able to determine if the services and service levels that are ultimately decided upon will meet the requirements to support the underlying business process.

Once the service provider has scoped the requirements and has determined costs to provide the services, the negotiation process can begin. It is important that both sides show flexibility in their approach to the negotiation process. Both parties should attempt to arrive at terms that satisfy their mutual needs. Inflexibility will not only drive up costs, but could jeopardize the entire negotiation process.

When the development team has reviewed the service provider's estimated costs associated with the team's proposed SLAs, they need to weight their requirements against the costs, and determine the services and associated quality levels that they can afford. Understanding the business impacts of the various levels of service is essential in this phase. The team must understand the minimum service requirements to support a business process, so funding is not wasted on satisfying requirements that are greater than necessary.

It is recommended that the type of services should be negotiated first, then technical issues, then legal terms, and finally price. (Sopko) When services, their associated service levels and costs have been negotiated, the remaining sections of the SLA detailing responsibilities, penalties, incentives, deliverables, documentation, methodology for verification, escalation procedures, and management of the SLAs will have to be mutually agreed upon. An important part of this negotiation is agreeing on the tools or products that will be used to monitor performance. Another area that must be discussed is the required reports, their format, their periodicity, and their distribution. Reports are extremely important in that they provide the mechanism by which management can determine whether actual performance meets service thresholds. The reports and other deliverables are usually outlined in the Contract Data Requirements List (CDRL).

SLAs also delineate areas of responsibility, which can make troubleshooting faults much easier. When a fault occurs, the SLAs can be used to achieve a team effort in which everybody understands their respective areas of responsibility. Poorly defined roles and responsibilities will lead to contractual challenges if SLA thresholds are violated.

Depending upon an application's criticality, and the services being offered, acceptance testing may be necessary. For example in a contract for hosting services, the application can be loaded on a server in the host facility and tests can be run to determine monitoring capabilities, resource utilization, software compatibility, and response times. Some vendors will object to this tactic, but the tests will ensure that the service provider can perform. It is not unusual for a service provider's sales staff to oversell their

capabilities in their zeal to close the deal. Acceptance testing not only ensures that the service provider has the technical skills to perform the service, but it establishes that the organization will be actively monitoring the contractual terms of the agreement. Depending upon the services being offered, the organization can run the acceptance test and maintain current operations in parallel. If the acceptance test fails, then it is easy to terminate the agreement. Details of the acceptance testing, including the methodology, tools needed, duration and associated costs will have to be negotiated.

8. Contract

When both parties are satisfied with the terms of the SLAs, the agreement needs to be formalized as a contract. It is important that everything that was agreed to is documented in the contract, especially termination and penalty clauses. It is also important that both parties agree to the terminology used in the contract.

The roles and responsibilities of each party should be clearly defined in the contract. The better defined the responsibilities are, the better the relationship between the two parties. Functions such as the method of communication, chain of command, points of contact and management of change need to be agreed to and documented. Additionally, issues such as who can place orders or modify requirements with the service provider, and what procedures are used to modify those requirements needs to be identified. In very dynamic environments it may be more important to manage the relationship than the contract.

Part of the negotiation process is to determine the scope and the duration of the contract. The scope clearly defines the services to be provided, and the boundaries for those services. The contract needs to specifically state those areas that are within the scope of the SLAs, and those services that are outside. For example, in a hosting services SLA, the service provider might not be held accountable for the latency experienced in the Wide Area Network (WAN) outside of the host environment. The scope also includes limitations such as the number of users supported, or application upgrades allowable. Availability services scoped for 100 users on the same infrastructure are very different from when the user base expands to 1,000 users distributed throughout the

country. Capacity requirements should be determined in the baseline tests, the service provider should not be held accountable for service levels when the application or the user base changes significantly.

The contract should also state the duration of the agreement. It is not recommended that a SLA contract be signed for more than a two-year period. Technology is changing too rapidly to be tied into a long term contract. In addition, the underlying business processes supported by the IT system can also be dynamic and rapidly changing.

The contract should also have provisions for review or revision of the SLAs. This is especially important if the development team was not able to capture good data on its benchmark analysis of the IT system. Often organizations have not adequately monitored their IT systems, so they are not sure of the level of service needed to support their business processes. As procedures are better defined, they may need to adjust the SLAs to reflect better defined requirements. Reports and monitoring tools may also need to be revised to better present the information to various levels of management and oversight personnel.

The service provider should also be able to address revisions to the SLAs. In many cases the service provider will not have the ability to conduct a thorough analysis of the IT system or application to be supported. Lack of due diligence may result in dependencies, resource utilization, bandwidth requirements and support that was not originally noted. Additionally disagreements on interpretation of the SLAs will have to be worked out. It is also possible that technical problems will force modifications to the SLAs, such as a particular monitoring tool that was agreed to will not interface with the application in the way it was intended.

SLAs are not static, as the workplace itself is not static. As experience is gaining in tuning and monitoring the application, SLAs will need to be modified or refined. Both parties should agree to modification of procedures and requirements as additional information is discovered regarding services provided and the efforts required to support those services. SLAs should be reviewed on a weekly basis for the first two or three

months. Any changes or modifications to the SLAs will have to be mutually agreed upon. The contract needs to be explicit in explaining the process by which modifications or refinements of the SLAs occur.

The contract also needs to discuss procedures to modify the SLAs because of changes as a result of application modifications, or configuration updates to supporting software or hardware. A mechanism such as a change review board must be instituted to address hardware or software changes initiated by either the customer or the service provider. The change review board should have membership from the program management team, the service provider, contracting representatives, end users, and possibly the business process owner. The change review board will review and approve software or hardware changes to the application or the supporting environment, determine if those changes will affect the SLAs, and if so, whether new SLAs should be agreed to. Changes that have not been approved by the change review board are unauthorized and the offending party will be held accountable. Additionally, the change review board should have a mechanism for identifying who should pay for additional resources (hardware, software, personnel) as a result of application changes, or changes to the system software.

Additional contractual provisions will have to be worked out if the nature of the application or its underlying business process is rapidly changing. This is especially true for prototype applications. Although the stability of the application should have been identified in the negotiation process, it is important that remediation processes are identified in the contract to account for rapid changes to the application. For highly dynamic applications or applications associated with businesses that must react quickly to external forces, mechanisms will have to be built into the contract to allow the contracting official and the change review board to quickly modify requirements and their associated service levels. SLAs are intended to protect business processes, not hinder them.

B. SUCCESSFUL SLAS

The method of developing SLAs as well as the formats of the SLAs may differ, but all good SLAs have similar qualities. This section outlines some lessons learned that

might assist in developing successful SLAs. The lessons are not presented in any particular order of importance.

- The SLAs should only focus on those requirements that drive a business need, or directly support a primary or secondary process. Focusing on the business need ensures the SLAs are meaningful, have management support, and can be justified financially. SLAs should be based on what is important to measure, not what is easy to measure.
- Service level agreements that measure the technical aspects of a service, yet fail to meet the requirements of the underlying business process will not be successful. Including the end users in the development process will help to focus on the customer's requirements.
- The number of SLAs should be kept relatively small. If there are too many SLAs, the service provider looses focus on what the mission essential service are, and monitoring and validating the SLAs will be more difficult and time consuming. Additionally, too many SLAs may deter good service providers from competing for the services. Too many SLAs will also prolong the negotiation process and ultimately cost the organization more.
- Robert F. Kennedy stated, "Progress is a nice word. But change is its motivator and change has its enemy." SLAs are only one part of quality control. The entire organization needs to be involved in quality management to achieve success. Upper management needs to implement the policies, drive the training, and allocate the resources to support the quality management initiative. Without upper management support, SLAs will not achieve the success they are capable of.
- Communicating the results of the SLAs to all of the stakeholders, in a timely manner is important. This is part of an organization's quality assurance effort to ensure that stakeholders have confidence in the quality of the services that they are receiving. A great deal of effort goes into developing SLAs, upper management should take the credit for initiating and managing SLAs as part of a quality control program. Both good and bad results should be shared. If SLA results are not being communicated, then stakeholders may believe that they are not being met, thereby eroding confidence in the ESP or the IT department.

- James Magory said, “computers can figure out all sorts of problems, except the things in the world that just don’t add up.” In other words, technology does not solve all management problems. SLAs should be used as part of a quality control plan, not as a tool to correct bad management. SLAs can identify where quality is not being provided, but SLAs will not solve the problem.
- Penalties or incentives must be used. Without them, the SLAs are just agreements. The penalties or incentives should not be too large, but they must command the attention of the service provider.
- SLAs must be easily understood by all parties. If the end users cannot understand the SLAs, then they are probably concentrating too much on the technical aspects of the service and not enough on supporting the business processes. Response times in a router mean little to the end user. The SLAs should reflect business terminology that the end users understand, such as the overall availability of the application, or mean time to failure instead of the listing the technical components that comprise the availability formula.
- If a service cannot be accurately measured, in a timely manner (enough to support the business process, which may include real-time), it should not be included as part of the SLA.
- SLAs should be reviewed frequently. SLAs will change, and they must be approached as a dynamic agreement. Change management processes need to be addressed in the development process, and agreed to in the negotiation process. Capacity planning is another area that needs to be addressed as new requirements may require additional resources.
- If prior service performance is not known, or if a new service is being initiated, trial SLAs without penalties or incentives may be necessary for a brief period (3 to 6 months). A cost-plus type of contract may also be helpful.
- A SLA is a contract and should be treated as such. To prevent any inconsistencies, the SLA and SOW should be negotiated as a whole. Most ESPs are very experienced in negotiating SLA contracts. They have the expertise; most organizations

do not. Organizations should not be afraid to bring in outside contractors experienced with negotiating service agreements to assist in the negotiations.

- If possible SLAs should reflect end-to-end services. It is important to look at the entire IT system. In a multi-tiered system, it is possible for all of the components to meet their individual availability thresholds, but when combined they still do not satisfy the end user's requirements. End-to-end SLAs are aligned more to the business processes they support.
- It is very important that both parties agree to terminology. For example, the term 'downtime' can be defined in many different ways. An ESP may consider 'downtime' to be when a server has a hardware failure, whereas the organization may consider 'downtime' to be when the end user cannot access the server from his or her PC. Unless the terminology is agreed upon, there will be many contractual issues. How will intermittent 'downtime' be handled?
- The SLAs or SOW need to address how the data and reports will be generated and stored. Issues such as who has access to the service level reporting tool, how information will be stored, and for how long, need to be discussed.
- Cascading SLAs can be a problem. This is when the service provider has to rely upon other third parties to perform a portion of the service being offered, and actions by the third party provider alters the original agreement. For example, service provider X may offer end-to-end SLAs to customer A. However service provider X has to rely on the long haul WAN services of provider Y. Service provider X and Y have a service level agreement for the long haul services. Service provider Y upgrades security protocols to meet the requirements of another customer. Service provider X must adopt the new protocol, which is not supported by customer A.
- End-to-end SLAs are difficult to achieve, especially in a highly distributed environment. Achieving high levels of availability for distributed applications requires control (physical or contractual) over the component pieces that make up the entire system and infrastructure, strict configuration control, proper monitoring tools, and a change control methodology that can adapt to rapid changes.

- SLAs are part of a quality control methodology. Once service levels have been measured and compared against the agreed upon thresholds, root cause analysis needs to be performed to determine why thresholds were violated. Once cause has been determined, the SLM organization needs to take the steps necessary to correct the problem.

C. POST-PRODUCTION SUPPORT

The SOW and thirteen SLAs in appendix (A) illustrate how SLAs can be used to improve the management and quality of software post-production support by establishing a monitoring program to support process and quality control measures. The SOW and SLAs in appendix (A) provide a detailed listing of post-production services and quality thresholds. A discussion of how those services and quality thresholds improve the management and quality of the software-intensive system would be redundant. Rather than focus on the specifics of how the SLAs in appendix (A) contribute to the quality of post-production services, this section will discuss how those SLAs were developed.

In the previous sections, we have discussed how SLAs should be developed, and offered some characteristics of good SLAs. In this section we will offer another approach at developing SLAs that will illustrate more of a top-down approach. The approach outlined in this section was utilized in the development of the statement of work (SOW) and SLAs in appendix (A) that were part of an actual request for proposal (RFP) to obtain quotes for post-production services.

1. Background

Today's computer environment differs significantly from the more centralized, mainframe-intensive environment of the past. Stand-alone and/or clustered servers have rapidly replaced mainframes as a result of the rapid adoption of the client-server architecture, the increased computing and storage capabilities found in today's servers, the dramatic reduction in server size, and dramatic drops in the cost of computer hardware. In addition, advances in distributed-computing technology, increased network speeds associated with broadband technology, and advances in web technology have also made the location of the server a moot issue. Low cost hardware coupled with

distributed-computing technology allows program managers to quickly purchase, configure, and deploy a system. Distributed-computing technology also increases the ease at which a program manager can outsource the hosting services associated with the application, as the server can be easily accessed using the Internet. While the current computing environment makes deploying a system easier for the program manager, it makes managing the applications, and servers more difficult at the enterprise level.

In an interview with a Chief Information Officer (CIO) staff member, he commented on the difficulty he was having tracking and managing servers. He said, "servers are worse than rabbits, I swear they are breeding. I am finding them everywhere, including under desks and in closets." Unless all of the IT funding is coordinated through the CIO organization, it is very easy for program managers to buy servers and deploy applications with little or no oversight. The proliferation of servers has caused numerous problems for IT departments.

One problem with the decentralization associated with servers vice mainframes is that it is difficult to standardize policies and procedures. Within the government it is not uncommon to find host service support ranging from twenty four hour support in a monitored hosting environment to servers that are receiving no support at all. This range of support can be the result of funding constraints where programs are trying to save funds by reducing the level of support. It can also be the result of a program manager's lack of technical knowledge.

It is difficult to manage post-production hosting contracts at an enterprise level unless the contracts are with a couple of stable, reliable contractors, and the services are similar. If program managers have the ability to independently contract with ESPs for hosting services, the range of services and quality requirements can vary dramatically. Even when host services are provided by an internal IT staff, services can differ due to varying business priorities, hardware differences, and obsolete operating systems necessary to support legacy systems.

Problems can also result when development was outsourced but hosting services were kept internal. Good communication is needed between the developers and the internal system administrators to ensure that network quality of service (QOS),

interoperability concerns, resource constraints, monitoring software, and security concerns are discussed and conflicts are resolved.

A third problem is that many of the program managers do not have the appropriate IT experience or background to be contracting for hosting services. Many of the program managers do not know what services are required to host their applications, nor do they know what levels of quality they should require in their contracts. In the government, contractors provide much of the technical expertise necessary to develop software-intensive systems. Some of these contractors are very familiar with the tasks necessary to support an application in post-production, but it is more common to find that the contractors specialize in particular areas of the development process.

Many of the larger IT consulting companies offer their own host services, which they include as part of the development contract. These contracts provide many of the services necessary to support an application, but the contracts are written to minimize the risk to the hosting organization. In most cases, the application is properly supported, but if problems occur, the host provider will have little if any liability.

2. Post-Production Services

The SOW and SLAs in appendix (A) were part of an effort by the Naval Supply Systems Command (NAVSUP) to consolidate their numerous servers, managed by multiple program managers and commercial entities, into a single hosting environment. As part of their server consolidation effort, NAVSUP wanted to explore the possibility of outsourcing hosting services. One of the sources considered was Electronic Data Systems (EDS). At that time the Navy was in the process of implementing the Navy/Marine Corps Intranet (NMCI), an effort to outsource all desktop and network support to EDS. Contract line item number (CLIN) 29 of the NMCI contract was written to include additional IT services, including hosting services. Although CLIN 29 was part of the negotiated NMCI contract, it was not priced, so the services provided under that CLIN had to be negotiated separately.

One of the security issues with NMCI was defining trusted boundaries. If EDS provided hosting services, the servers would be within the NMCI trusted boundary, offering greater security. Any other service providers would be outside of the trusted

boundary, and access to those services would have to travel through the NMCI external firewalls. Outside access would require greater security restrictions at the external router and firewall (e.g., port restrictions and protocol restriction such as use of Active X). As such, the Navy was exploring the option of having EDS provide hosting services as part of CLIN 29.

Since hosting services under CLIN 29 had to be negotiated, the author was tasked by NAVSUP to develop a contract for hosting services. Since the NMCI contract had already been awarded to EDS, the author was able to negotiate a hosting contract with EDS that would provide the services necessary to support NAVSUP's applications, contained enough flexibility to meet the requirements of specific projects, and was capable of being performed by EDS. The author presented initial requirements and SLAs to EDS. The resulting SOW in appendix (A), was a collaborative effort between the author and EDS (specifically Scott Price and Joe Vickery). The final product of the SOW and SLAs were written to augment CLIN 29 of the NMCI contract, so they could be used by any Navy activities requiring hosting services.

The SOW and SLAs contained in appendix (A) were intended to provide a listing of services and service levels that the program manager could use in outsourcing contracts, or in negotiations for support with an internal IT hosting provider. Appendix (A) provides thirteen SLAs and three levels of service, which should contain sufficient options for most programs. Although the SOW and the SLAs in appendix (A), are intended to be used as a template to be modified to meet specific needs of an application.

At the time of this writing the NMCI program office had not accepted the SOW and SLAs as part of the NMCI contract, although working groups were formed to further define CLIN 29. The work in appendix (A) was provided to the group for their consideration. There are numerous business and political reasons for not immediately adopting the work in appendix (A), but due to the sensitive nature of these issues they will not be discussed in this dissertation.

The SOWs and SLAs were however, used by NAVSUP to contract for server hosting services. Although two commercial entities bid on the work, and a source selection board was convened, the contract had not been awarded at the time of the

contract. Again a detailed discussion of why the contract was not awarded will not be discussed due to the proprietary nature of the bids and the sensitivity of the information. However, the failure to award the contract was not attributed to either the SOW or the SLAs.

3. Developing the SOW and SLAs in Appendix (A)

The first step in developing SLAs is to define the problem that needs to be solved. In this case the problem was that NAVSUP wanted to consolidate their servers under one hosting service provider. NAVSUP needed to generate a requirements document that listed the services and service levels necessary to support its applications. Ultimately, these requirements were to be used to form a proposal under CLIN of the NMCI contract.

Although recommended, a team approach was not utilized in the creation of the SLAs in appendix (A), although the SOW was formed with a small team. Before a team was formed, we conducted an initial inquiry to determine the services that program managers needed to support their applications. Initial interviews and inspections revealed that there were no standards or procedures for application hosting. While almost all of the applications were receiving adequate services, the services and service levels varied greatly. Mission critical systems received good support, while those programs struggling for funding provided little support. The disparate services being provided, and difficulty gathering program managers and stakeholders for a SLA development effort did not allow for a good bottom-up approach to developing the requirements. A better approach was for the program managers and stakeholders to validate a list of services and service levels that were derived from a top-down approach.

The top-down approach consisted of the author determining which hosting services and service levels were necessary to support an application. The personnel requirements and activities associated with SLM were assumed as some of the personnel that were displaced as a result of any outsourcing were going to fill needed SLM positions. The initial requirements were developed from a review of the hosting services that were being performed at that time. Requirements were also derived from conducting benchmarking studies, reviewing previous contracts, literary searches (Philcox, Nemeth,

Minasi, Sjouwerman, Harney, OGC, and Factor), interviews, and collaboration with EDS personnel. The resultant product formed the initial requirements generation document.

As was mentioned previously, SLAs have been used for a number of years. However, a review of many commercially provided SLAs and those contained in previous contracts were ambiguous, difficult to measure, lacked qualitative measurements, or lacked penalties/incentives. The SLAs also lacked many of the elements that we felt were necessary to address in both the development of the SLAs and the enforcement of the SLAs.

The author attempted to address many of these deficiencies by writing the SLAs utilizing a new format that required more information on the services being performed, how those services will be measured, and the responsibilities of all parties. The author also attempted to use the SLAs as a process and quality control mechanism to assist the program managers in the performance of their oversight duties. As such, the author also had to develop additional quality requirements for security, documentation, maintenance, tape backups, and technology refresh. These requirements were derived from prior experience, interviews, literary searches, review of current services, and prior contracts.

Once the initial requirements were gathered, the author met with EDS to assist in the development of the SOW. It was decided that the majority of the programs evaluated could be grouped into three packages of services (essential, enhanced, and premier). After much collaboration, the services were grouped into one of the three categories. Although most of the programs could be adequately supported by the services in the essential package, some programs required additional services due to their mission criticality. Once the services were grouped into the three packages, the SLAs had to be modified to reflect three levels of quality thresholds. The SLAs were also reviewed by EDS and were modified to increase readability, reduce ambiguity, incorporate better monitoring capabilities, and reflect penalties that were within the range of compromise (penalties are not designed to financially cripple an organization, they are designed to entice an organization to comply with requirements).

The final product was presented to program managers, the NAVSUP CIO staff, System Administration personnel, EDS management, and two IT consulting groups for

their feedback. Their responses were very favorable, but a common concern was that the services would be too expensive.

Once comments concerning the SOW and SLAs were addressed, NAVSUP decided to utilize seven programs in a Request for Quotation (RFQ) utilizing the SOW and SLAs. The RFQ was given to EDS and one other activity. Although details of the proposals cannot be discussed in this dissertation given the business sensitivity of bids, general impressions from the source selection board and the two organizations involved was very favorable. The companies liked the level of detail contained in the SOW and SLAs, although they did not like the penalties associated with non-performance. The program managers also liked the comprehensive list of services that were being offered; in many cases they had not though to include some of the services in their own contracts. The source selection board indicated that due to the level of detail contained in the SOW and SLAs, they were able to better compare the services offered by the two organizations. They were able to disregard services (in many cases marketing hype) that were offered by the companies, but were not contained in the SOW. This allowed a better "apples to apples" comparison.

Although a contract for host services was not initiated for these seven programs, the SOW and SLAs in appendix (A) were still being evaluated for inclusion as part of the CLIN 29 of the NMCI contract. Due to political and business sensitivity, and the possibility that the source selection between EDS and the other organization is still a possibility, the author felt that it was more appropriate to use generalities in this discussion.

D. SUMMARY

Many of the benefits from SLAs are derived from the process of developing the SLAs. The development effort is best when a team approach is utilized, where each of the stakeholders is represented and has input. When the team members feel that they are a part of the process to improve the software quality, they are more likely to take ownership of the quality assurance and quality control processes established.

One of the major benefits of developing the SLAs is improved communication between all of the stakeholders. The SLA development team identifies critical business

processes and jointly determines the quantifiable quality factors necessary to support the process and meet the organization's needs. The team must also determine the means to determine whether quality factors have been met, which encourages communication with the test community. Developing the SLAs fosters a common understanding about quality and performance requirements across the organization. The SLAs also explicitly state the quality thresholds, which helps to limit unrealistic expectations by management and the end users.

The SOW and SLAs in appendix (A) demonstrate how SLAs can be written to improve the quality of post-production services. The SLAs establish many of the quality and process control measures that program managers need to properly manage post-production support. The SOW and SLAs in appendix (A) incorporated three levels of service to satisfy the majority of program needs, but they could be easily tailored to meet the specific needs of a program. The SOW and SLAs also helped the average program managers by detailing services and quality thresholds that they many not have thought of. Appendix (A) offers a good template that other program managers can utilize in their software acquisitions and post-production support contracts.

The discussion outlining how the SOW and SLAs in appendix (A) were developed illustrates some of the difficulties associated with software acquisition. It is not always possible to get all of the stakeholders together for a development effort. In the case of the SOW and SLAs in appendix (A), a top-down approach, which was later validated by stakeholders proved to be the best approach.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SOFTWARE DEVELOPMENT MODELS

One of the original contributions of this dissertation is to apply SLAs, which are used primarily in post-production support contracts with ESPs, to the entire lifecycle of software development in an effort to increase the management of the development process, or increase the quality of the deliverable or output from the development process. However, before we explain how SLAs can positively affect the various lifecycles of the software, those lifecycles bear discussion.

Models have been used for a long time to describe work processes by utilizing a top-down approach of decomposing the processes into discrete sets, then showing how information flows among them. (Nutt) Process models abstract the real world into sets of entities that flow through a system of activities such that they can explicitly capture the process artifacts, and information flows. (Martin)

Software production is an extremely complex process. The complexity stems in part from the difficulties in comprehending the various facets of the design problem in order to derive a robust and reliable design. (Ewusi-Mensah). One means of reducing the complexity is to develop a model that describes a set of activities (in sequential order, recursive, or conducted in parallel depending on the model) that needs to be accomplished to produce a software product that meets requirements. Presenting the development process in an abstraction allows a better understanding of the tasks to be accomplished, as thus assists in the selection of the proper methods and tools to accomplish those tasks.

Early developers modeled their processes in an attempt to improve software development and product quality by applying a systematic development process based on lessons learned from other software development projects. As new tools, procedures, and lessons emerged; new process models have been developed. These software process models are used to guide development efforts by outlining a deliberate set of activities at an appropriate level of abstraction to create a software product that addresses end-user requirements.

In addition to providing a strategy to address the requirements initially proposed in the Statement of Work (SOW) or Performance Work Statement (PWS) process models can be utilized for a number of purposes. Process models form the basis for planning, organizing, staffing, budgeting, scheduling, and directing software development activities. The models also can be used for analyzing or estimating resource requirements, determining what software engineering tools and methodology will be most appropriate to support various development activities, and providing a basis for empirical studies to analyze and evaluate the effects that the prescribed activities had on cost, schedule, and performance. (Scacchi) Software process models are also useful in contracting to identify milestones and deliverables. Test plans can then be developed to evaluate the deliverables for conformance to stated product and quality requirements.

A. TYPES OF PROCESS MODELS

Every software development effort follows some process. The development process may be informal, ad-hoc (some prefer the term chaotic), or it may be well documented with procedures to actively monitor the process. There are numerous process models that range from general models at high levels of abstraction to models that are specific to a particular domain and are at a very granular level.

Over the last three decades there have been numerous software process models developed in an attempt to improve efficiency, effectiveness, and quality in the development process in an effort to improve quality in the end product. The models differ in their approaches, methodologies, level of abstraction, relevance to the real world, structure, and incorporation of variables, such as user interaction. The models also differ in techniques used. The models can incorporate modeling techniques such as data modeling, object modeling, entity diagrams, process programming (uses programming notation and formalism to model the process), precedence networks, or Petri Nets (formal mathematical notation). (Gibson)

There are numerous ways to categorize process models. There are also as many ways in which the models are utilized (i.e., some models estimate effort or duration, others analyze risk, some are intended to improve quality, and others are intended to

improve documentation). Three methods of categorizing software process models are presented below. It is also important to realize that the software product lifecycle can be viewed from a number of perspectives. Albin describes four perspectives on how management, software engineers, software architects, and software designers can view the same development model differently. (Albin)

Martin and Raffo broke software models into two categories. (Martin) The first category included models that estimated development characteristics (e.g., quality, duration, effort) by identifying key variables and determining their effect on the development process. Examples of this type of model include COCOMO II (Boehm, 2000), and Software Lifecycle Management (SLIM) (Putnam). The other category of models attempts to estimate development characteristics by modeling and analyzing the details of the development process. Examples of this approach include models by Raffo, Harrison, and Vandeville (Raffo), the Software Engineering Institute's Software Capability Maturity Model (SW-CMM) and Personal Software Process (Humphrey).

Schacchi also broke the process models into two categories, but each had several subcategories. (Schacchi) The first category was software lifecycle models which included those models that provided a framework to organize and structure how software development activities should be performed and in what order. Subcategories included Classic Software Lifecycle models (software evolution proceeds through an orderly sequence of transitions from one phase to another), Stepwise Refinement (systems are developed through progressive refinement of high-level specifications (requirements and design) into more concrete low-level specifications capable of being converted to code), Incremental Development and Release (development consists of providing core functionality, then incorporating new requirements for an improved release), Industrial and Military Standards (these include the CMM models, which provide standardization of procedures and deliverables), and a subcategory called alternatives (focuses on the product, product processes, or production setting) which includes models such as rapid prototyping, joint application development, and component based development.

The other category was software production process models, which are models that represent a networked sequence of events, activities, objects and transformations that

form a strategy for accomplishing software evolution. These models use rich notation, syntax and semantics to develop more precise and formalized descriptions of software development activities. Schacchi broke the software production process models into two subcategories of operational and non-operational models. Operational models can be viewed as computational scripts or programs, where many of the processes are automated within a software language or tool. These models take a formal specification and generate code, which can constitute a functional prototype. The code can also be analyzed for certain characteristics and parameters. Many of the fourth generation techniques (4GT) are operational models. Non-operational models present conceptual approaches to development, but they have not been developed to the point where they can be automated or codified. He sites the Spiral model, (Boehm, 1988) as an example of this type of model because it incorporates elements of specification and prototype process with a traditional lifecycle model. (Schacchi)

Pressman broke software process models into 7 different categories. (Pressman) The first category was linear sequential models, like the waterfall model, that defines development activities and illustrate a sequential process to execute those actions. Another category is prototype models, which include iterative steps of defining requirements, designing the system, developing a prototype to test the concept, revising or enhancing requirements and repeating the process until a final product is developed. Rapid Application Development models are another type of process model where very short development cycles are utilized to quickly develop specific functionality in a system, modules within the system, or if the project is small enough, the entire system. Another large category of models is Evolutionary Software Process Models, which are iterative in nature. These models provide an initial release, then add or enhance functionality. Formal Models are another category in which formal mathematical specifications are used to apply more rigor. The last category includes those process models that incorporate fourth generation techniques, which include automated activities that translate specifications into source code.

B. SELECTING APPROPRIATE PROCESS MODEL

Software products are unique. Requirements, resources, budgets, personnel, interface requirements, and external influences are never constant from one project to the next. As a result it is better to think of software as being developed rather than produced. As a result, software process models have to be tailored to meet the specific needs of a particular project. (Verlage) The choice of development model (including tailoring for a particular project will depending upon the specific performance dimensions (e.g., defect rates, KLOC produced per day) that must be optimized. (MacCormack)

Selecting the appropriate process model is one of the most important activities in the project development planning effort. The appropriate model can streamline a project, maximize resource utilization, systematically ensure that activities are accomplished to achieve stated objectives, satisfy user needs, increase tracking and control, minimize risk, and improve quality. Conversely, the wrong process model, or no process model can result in longer schedule times, rework, unnecessary work, poor requirements, and frustration. (Alexander, McConnell) While choosing an appropriate process model is important, is should be noted that adhering to specific processes does not guarantee a successful project.

Software development models used today vary in approach, methodologies, domains of interest, areas of development, and level of abstraction. Given the large number of software development practices and models, selecting the right mix of practices and models is difficult. It is not possible to find a single model that will incorporate a set of practices that will optimize performance on all dimensions. As such, program managers must tailor the process models to each project's specific requirements.

At the beginning of a project, the program manager should determine the primary performance objectives for the software deliverable, as those objectives will drive the type of development model utilized as well as the mix of practices they should utilize. (MacCormack) A software development model should also be selected based on the nature of the project and application, the methods and tools to be utilized, the controls and deliverables that are required, and the application domain. (Pressman)

The IEEE Standard 1074 (IEEE Standard for Developing Software Life Cycle Processes) outlines the activities necessary to develop software processes specific to a software project. The first step is to select a software lifecycle model (IEEE Std. 12207 describes 4 models and IEEE Std. 1012 describes Boehm's Spiral Model). Once a model is chosen they must be tailored to the project at hand. This activity is described as mapping where the project-specific sequence of activities are selected or added to the software lifecycle model. The result of the mapping is the project software life cycle. The next step is to evaluate an organization's environment (policies, standards, tools, procedures, and metrics). When the organization's environmental variables are incorporated into the project software life cycle, then the software life cycle process is determined. (Schmidt)

Eljabiri and Deek describe software process models as a problem-solving framework designed to solve real world problems, within time and resource constraints. They identify a number of factors that have influenced the evolution of software process models. (Eljabiri) These same factors also need to be evaluated when selecting a process model to ensure that the model is accounting for the relevant factors. One of the factors they discussed was the time dimension of the project (i.e., the anticipated length of the project). The length of the project impacts other variables such as requirements churn resulting from environmental change, the degree of visualization, complexity, software economics, and changes in technology. Projects with a long development cycle should select a different and more flexible process model than projects with shorter cycles.

Other factors that need to be considered include the amount of automation, the degree of control required/desired, the degree of interaction with other systems, and experience with the development process proposed. Eljabiri and Deek also identified the importance that cognitive psychology had on process models. Behavioral models, use-case approaches, and prototyping are effective strategies if requirements are not well known, or if there is organizational conflict concerning requirements.

Alexander and Davis also presented guidelines for selecting the appropriate software process model. (Alexander) They described 20 criteria that they felt could be utilized in selecting the most appropriate software process model for a specific project.

They selected three grades for each criterion, and evaluated a number of software process models to determine whether the model satisfied the criterion at any of the three grades. To determine the best model for a particular project, each criterion would be graded based on the characteristics of a particular project. The model with the highest ranking (satisfied the most criteria) would then be selected.

The criteria were broken into five categories, each containing sub-categories. Each of the sub-categories was scored using three values that corresponded to the type of sub-category. The categories were personnel, problem, product, resource, and organization. The category of personnel was further divided into user experience in application domain (corresponding values for experience were novice, experienced, and expert), user's ability to express requirements, developers experience in application domain, and developer's software engineering experience. The category of problem was subdivided into maturity of application, problem complexity, requirements for partial functionality, and frequency of change. Product category was subdivided into product size, product complexity, non-functional quality requirements, and human interface requirements. The resource category was broken down into funding profile, funds availability, staffing profile, staff availability, and accessibility of users. The organizational criteria were subdivided into management compatibility and quality assurance/configuration management. Matching the appropriate project criteria against the variety of models to determine which model best meets the program manager's needs is an important part of the problem solving process required in software development.

C. PROCESS MODELS

As mentioned earlier, there are numerous software process models, including MIL-STD-2167-A, the Rapid Prototyping model, the WinWin Spiral model, ISO-12207, Incremental Development and Release model, the Component Assembly model, the Concurrent Development model, the Cleanroom model, hybrid models, object oriented models, and fourth generation models. This section will describe some of the process models and will point out the advantages and disadvantages of using each.

A common model is the early days of software development was the code and fix model. In this model the developers have a general idea of requirements, then they use a combination of methods to code and debug the software until they have a final product. This approach has the advantage of low overhead (little effort on documentation, standards enforcement, quality control), and anyone can use this model, as it requires little or no experience. This approach can be useful for very small projects with a well-defined solution space, a proof of concept, or throw away prototypes. (McConnell) Despite its obvious faults, this model is the most common of all software development methods, as it is the default model if no other process models are utilized. (Charvat)

1. Waterfall Model

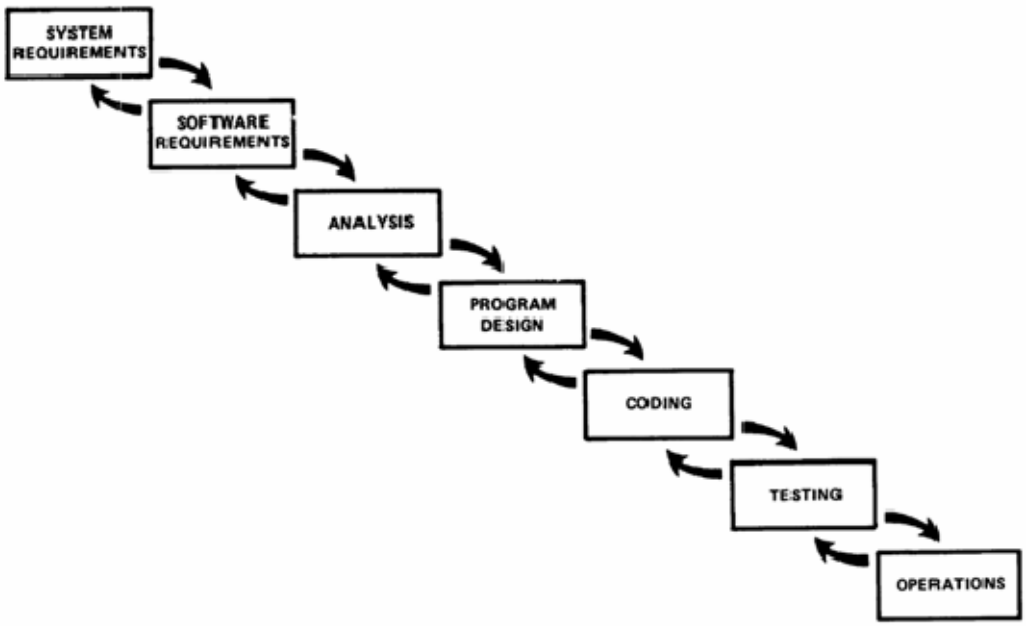
The waterfall model (Royce) was the first attempt at formalizing the development process by identifying an ordered set of work steps. (Becker) The waterfall model is a sequential software process model that was based on traditional industrial engineering techniques. Despite the fact that it was developed in 1970, it still serves as the basis for many, more effective software process models. (Eljabiri, Rakitin)

In the waterfall model, development starts with the initial concept for the software-intensive system and progresses through a sequence of phases until the system undergoes testing and is approved. The phases or steps do not overlap. Each phase is dependent upon the products produced in the prior phase. The waterfall model also contains transition criteria for progression from one stage to the next. Only when a deliverable or documentation is produced for a specific phase, and is approved by the program manager, can development continue to the next step. If a deliverable is not complete, then the project must remain in the current phase until the deliverable is made acceptable. If an error is discovered at some point in the process it is possible, although difficult, to return to an earlier step.

The model begins with understanding the requirements for the entire system. Functionality is then assigned to hardware and the software components. Software requirements are generated, documented, and in many cases modeled. The requirements are then analyzed for accuracy, consistency, conflicts, level of detail, amount of information, and adherence of overall system requirements. The deliverable from this

phase is the software requirements specifications, which are then used by the software programmers to develop the software design. In the design phase the software architecture is developed and functionality is assigned to the various software components or modules. The documentation from the design phase is then used by the programmers to translate the requirement specifications into code. The test phase validates that the coded software meets defined requirements. When the software has completed testing and has been approved, it is then released to its intended customers and the operations phase begins. As maintenance activities are required, the model begins anew.

The waterfall model is still popular in that it is easy to understand, it has well defined deliverables at the end of each stage, and it emphasizes requirements analysis (define before design, design before code). (Rakitin) The waterfall is a rigid model, but it works well when requirements are well known, the technology is mature, and developers are experienced.



FIGUR E 2. WATERFALL MODEL

The major disadvantage of the model is the assumption that once requirements are defined that they will not change. As such the model does not reflect the true iterative

nature of development and requirements churn, therefore, it is rarely adhered to in actual use. Another disadvantage is that testing is conducted too late in the process to prevent problems. Despite its major disadvantages, the waterfall model is still widely used.

2. Spiral Model

Instead of the traditional document-driven or code-driven process models, the spiral model was an evolving risk-driven model. (Boehm 1988) The spiral model is broken into four quadrants: planning, risk analysis, development and assessment. The spirals through the various quadrants represent increased costs. Each cycle of the spiral begins with requirements engineering, analysis and selection of alternative methods of implementation. The purpose of the system or software component is determined with respect to functionality, quality attributes, and performance. Alternative methods are then determined (COTS, reuse, different designs), and constraints are identified (cost, schedule, interfaces, resources). The next step is to evaluate the alternatives in respect to the requirements, constraints, and risks. Part of this step is risk mitigation by identifying areas of uncertainty and collecting more information, or by developing prototypes, simulations, or conducting benchmark studies. The next step depends upon the risks identified. During the first spiral many of the risks involve requirements, so efforts are made to improve and refine requirements. As the spirals expand outward, the risks associated with the development effort increase, and detailed designs of the system are developed. The last step in the spiral is planning for the next level of prototyping or development of a more robust design. As requirements become more defined, and program development risks dominate, the steps will start to follow an incremental version of the waterfall model (requirements determination, design, code and test). (Boehm 1988)

The spiral model has a number of advantages over more traditional models. The largest advantage is that it represents the real world iterative approach to software development. It also incorporates the best of the waterfall model (stepwise approach) and the rapid prototyping model. The model also demands a risk assessment (requirement as well as technical risk) at each stage within the spiral. The risk mitigation focus of the model as well as emphasis on prototypes, simulation, and benchmarking, if properly applied should reduce risks before they become problematic. (Pressman) The spiral

model also has some disadvantages. The major disadvantage is that it requires considerable risk assessment expertise. It is not a widely used model as it is difficult for managers without a technical background to understand. It is also difficult to convince customers that an evolutionary approach with multiple prototypes is cost effective, controllable, and fast enough to meet market demands. Another disadvantage is that the model is risk based, so if a major risk is missed, problems may result. (Rakitin) Although performance and quality requirements can be addressed with risk analysis, the model does not specifically address those issues, so it is incumbent upon the users or the developers to include those areas in the risk assessments. (Schmietendorf) A final critique is that it can be difficult to define verifiable milestones that indicate whether a program is ready to proceed to the next layer of the spiral. (McConnell)

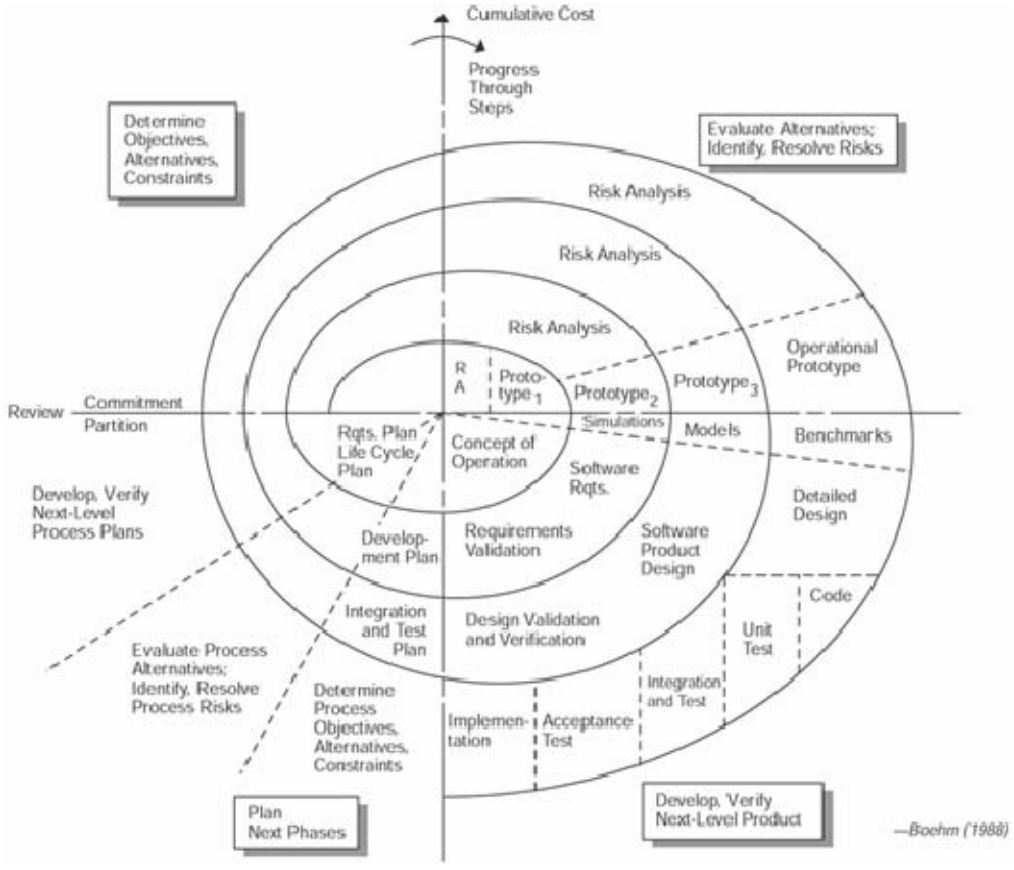


FIGURE 3. SPIRAL MODEL

3. Evolutionary Prototyping Model

There are a couple of models that are considered evolutionary prototype models. These groups of models have similar characteristics. These models develop the system concepts and requirements through the various iterations or evolutions of the model. The models begin with requirements elicitation and analysis. The developers try to capture the most stable and visible requirements. They design and code that portion of the system as a prototype, test it for functionality and conformance to stated requirements, and show it to the customer. After customer feedback and additional requirements engineering, the developers begin another iteration of the development cycle, adding additional functionality to the prototype. This process continues until the users determine that the system is "good enough", at which point it is released. (McConnell)

Figure 4 shown below, from Wiegers' book on software requirements (Wiegers), presents a model that incorporates three types of prototypes. Vertical prototypes are designed to function like the actual system at a specific structural level. Vertical prototypes act as a proof of concept to ensure interfaces function, algorithms perform to expectations, or architectural approaches are sound.

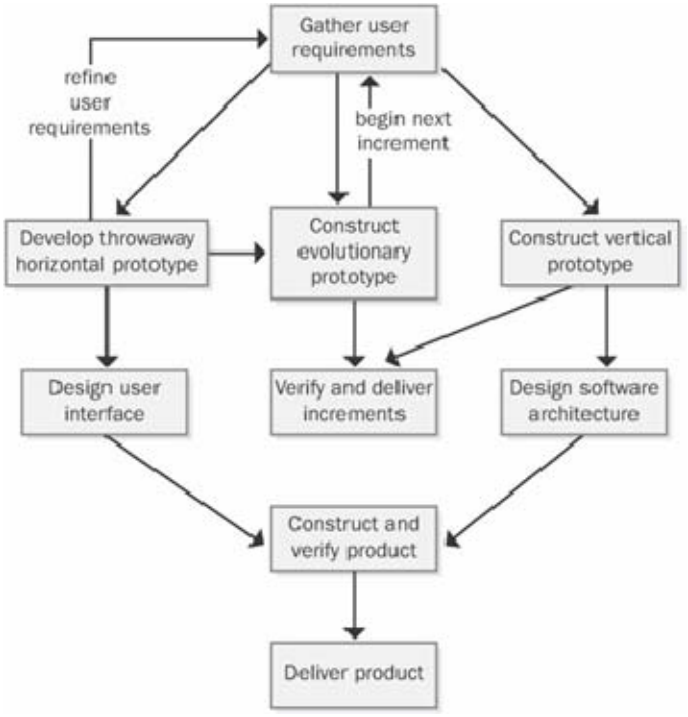


FIGURE 4. EVOLUTIONARY PROTOTYPE

Horizontal prototypes are used primarily to demonstrate portions of the system to the user. These types of prototypes show some functionality (e.g., graphic user interfaces, screen layout) without the actual implementation. The evolutionary prototype differs from the other prototype types in that it provides a solid architectural foundation for building the software incrementally as the requirements become better defined over time. (Wiegers)

In this model the developers can utilize several approaches to refine requirements. Horizontal throwaway prototypes are being used to refine user interfaces, while parallel efforts utilizing vertical prototypes test concepts. Both prototypes feed back into the evolutionary prototype, which also goes through a number of iterations until the final product is delivered.

The advantages of this model are that many of the processes occur in parallel, the model has stepwise refinement and multiple iterations to reflect real world experience. Some of the disadvantages to this type of model include determining when a project is "good enough" to deliver to customers, documentation and configuration management is a challenge, and it is difficult to keep the same stakeholders engaged in prototype evaluations through multiple iterations.

4. Commonality Among Models

Most of the software process models have the same basic activities, although the order of the activities, the iterations through the activities, and the deliverables associated with the activities differ. The models all begin with an evaluation of the system to be built. The project may be adding functionality/updating technology on an existing system, or it may consist of building a new system. System requirements are then broken into components and functionality is assigned to either the hardware or software. The software requirements are derived from the system requirements.

Another group of activities can be grouped into requirements engineering activities include defining stakeholder needs, business objectives, system functionality and performance parameters, resources, and constraints. Prototypes are often used as part of this activity to refine or capture user requirements. This activity also includes requirements analysis to ensure the requirements are not in conflict, that they are

complete and quantifiable. The requirements are then gathered and incorporated into specifications, which document the requirements.

An additional series of activities involve design. The requirements will specify what they want the system to do in terms of system behavior and performance. The designers will determine how the system will meet those requirements. The designers start by identify objects of computation, their attributes and relationships, operations to transform the objects, and constraints on system behavior. Then they divide the system into components denoting logical subsystems. These components can then be evaluated to determine if existing software already exists that can meet requirements (software reuse, component-based engineering, object-oriented designing), or whether new software will be needed. The architecture design is also conducted to define the interconnection, and resource interfaces between subsystems, components, and modules. Detailed component design then determines the means that specific modules will transform inputs into outputs. (Scacchi)

Coding is the activity that transforms the design specifications into actual source code. As the code is completed for each module, it is packaged into the overall system software. As errors are discovered in either the module or interfaces between components or modules debugging efforts are performed to correct the code.

Testing is another activity. In some models the testing validates the final deliverable, while other models conduct testing to validate the deliverables at each stage of the model. The goal of testing is to discover errors, validate design, and verify conformance to user requirements. All models conduct some form of testing at the unit level, module level, subsystem level, system level, or a combination of levels. In addition to evaluating the code or design, testing is also used to verify and validate other deliverables such as documentation.

The final activity is the post-production deployment of the system. This action consists of documenting the system (user guides, installation instructions, configuration documentation, system support information), installing the system in its host environment, configuring access, tuning the application, and performing system backups. This activity also includes training the end users, management, and system

administrators. The final activity is maintenance of the system, which includes repair of the existing system, modification of the system, and rehosting of the system.

D. SLAs AND SOFTWARE PROCESS MODELS

Many of the software process standards are based on the assumption that following a defined engineering process and having a quality management system, that higher quality software can be consistently produced. (Gibson) This is not necessarily the case. Despite claims that adherence to a specific process model improves software quality, the data to support most models is anecdotal and biased towards reporting only successful projects. (MacCormack)

Software models should act as guides. High-level models should be interpreted as an expression of general intent. (Nutt) Strict adherence to the models will result in problems as a model's abstractions hides many of the problems and tasks that must be accomplished at lower level design. Real world problems such as incomplete and changing requirements, unplanned dependent activities, time constraints, and design rework as a result of discovery can force organizations to deviate from planned processes. It can also cause inconsistencies between high-level processes and those that are more granular.

Software process models describe the sequence of activities necessary to produce a software product, processes involved, tools necessary to perform those functions, and exit criteria (deliverables) for moving from one activity to another. However, these models are abstractions, and thus, do not capture some of the important variables that can impact program success. For example, many of the models do not deal directly with performance or non-functional requirements, and if they are addressed it is only indirectly and without systemic background. (Schmietendorf) Few models, if any, focus on representing organizational goals and process improvement. (Turk)

Although, there are some models that address software program management activities in the process model, such as Abdel-Hamid and Madnick's model, which simulated the effects of staffing delays, schedule pressure, and unplanned work (undiscovered errors) on a projects' planned cost and schedule, and Boehm's Spiral

Model which included risk analysis, no model incorporates a holistic view of software development management. (Abdel-Hamid, Martin)

There are good software process assessment model such as CMM and PSP that measure how well processes are defined and adhered to, but they do not specify which processes are most appropriate, nor do they evaluate the quality with which the processes are executed. The focus on these models is on process management and process control, not on process quality or development quality. Critics have complained that approaches such as ISO 9001 and CMM emphasize managerial tasks and ignore the more important technical considerations.

Most of the software process models lack quality control and monitoring methods. All process models have transition criteria for progressing from one phase, activity, or module to another. The models typically have completion criteria for a current phase, and entrance criteria for the next stage. (Boehm 1988) Due to the abstract nature of the models and the recognition that the models will need to tailored, quantitative parameters for criteria acceptance is not specified. As a result, additional tools are necessary to compliment the software process models at the practical implementation level. SLAs are one of those tools.

SLAs can be used with any process model in an attempt to incorporate process control and interject quality and performance requirements into the completion and entrance criteria for the various stages of a process model. In the requirements engineering phase of the development process SLAs help to identify non-functional quality and performance requirements that are usually not considered until later in the development cycle. In the design phase, SLAs can be used for process control and to ensure the deliverables meet stated quality requirements. SLAs can also be written to monitor and evaluate a contractor's compliance to agreed-upon processes, methods, standards, tools, and procedures. SLAs can assist testing by identifying quantitative quality requirements for the deliverables at each phase of the process model. In the post-production phase SLAs can be used for process control and to identify the quality requirements necessary to ensure the application is properly supported.

As was previously mentioned selecting the appropriate process model is one of the most important activities in the project development planning effort. The selection of the appropriate process model is based primarily on the project's primary performance objectives. SLAs assist the selection process by identifying performance and quality objectives in addition to functional objectives. Non-functional requirements may well require a different process model than if only the functional characteristics of the software were considered.

The remainder of this dissertation will demonstrate in more detail how SLAs can be utilized at the various phases of software development to establish performance and quality requirements for deliverables as well as establishing monitoring actions to measure process compliance and detect problems through all of the major development steps. The dissertation will also demonstrate how SLAs can also be utilized to assign quality parameters to many of the management processes and activities associated with software development.

E. SUMMARY

This section was intended to illustrate the numerous approaches to developing software. Although there is a lot of commonality, each model represents a unique approach to development, including different processes, methods, and tools. Additionally, the need to tailor the models makes strict comparisons of models even more difficult. However, regardless of the development model selected, SLAs help to establish quality control measures by defining quantifiable quality thresholds for the deliverables expected at the various steps. SLAs also help to establish a process control program to measure adherence to whatever process is selected.

THIS PAGE INTENTIONALLY LEFT BLANK

V. REQUIREMENTS ENGINEERING

The first step in the software-development process is the requirements engineering process, which entails those activities necessary to determine a system's functions, capabilities, and behavior in order to satisfy the customer's needs. Requirements engineering is a process of discovery, refinement, modeling, specification, and validation. (Pressman) Skilled requirement engineers, management and stakeholder commitment, time, and proven processes are needed to deliver a good product. Requirements engineering can be very difficult, but the level of effort dedicated to requirements engineering will have a direct impact on software quality.

Requirements engineering provides the building blocks for all other efforts in the software engineering process, so if quality is not addressed at the beginning of the software engineering process, it is usually addressed at the end of the project in the form of testing. Unfortunately, quality evaluations are usually implemented too late, and the architecture that was already developed will dictate the solution space for addressing problems that were discovered.

Unless the requirements engineering process is performed correctly, there will be an expectation gap between what the developers though they were supposed to build, and what the stakeholders really needed. (Wiegers) Errors made in the requirements stage account for 40 to 60 percent of all defects found in a software project, yet organizations still practice poor requirements engineering processes. (Weigers) This chapter will discuss the requirements engineering process and demonstrate how SLAs not only improves the requirements engineering process, but how they help to inject quality requirements into the beginning of the development cycle.

A. SYSTEM REQUIREMENTS

Software requirements engineering begins with the overall system requirements, as specified in the requirements engineering portion of system engineering. Before software can be developed, the requirements for the system in which the software resides needs to be defined at some level of abstraction. When developing a new system,

requirements engineering is used to determine the customer's needs. The process starts with the customer developing an initial problem statement. Users, program managers, and system engineers need to determine how the system must behave to support the overall objectives of the system. From that problem statement, the system engineer must determine the product's mission, functionality, performance levels, availability, design and interfacing constraints, information needs, communication needs, and other system specifications. System engineers then need to scope the system; identify the roles of hardware, databases, and software; factor in user interaction; define processes and data sources; identify constraints; and determine interfaces with other systems.

Requirements engineering consists of requirements elicitation, requirements analysis and negotiation, requirements specifications, requirements validation, and requirements management throughout the development process. (Pressman, Weigers) One of the main objectives of system level requirements engineering is to determine which components will be used to satisfy specific requirements. (Sawyer) Will a specific requirement be satisfied through hardware, software, or a combination of both? When system level requirements are defined, the software engineer will then go through similar steps to convert system, user, and program requirements into a software design specification that developers can use to start coding.

This chapter will discuss requirements in the context of software requirements; however, SLAs can assist and play a role in system engineering as well. It is important to keep in mind that software is but one portion of a greater system. The quality standards that are determined through the systems engineering process will flow down to the software requirements. Software quality must be considered in the context of the entire system, not just the software.

B. REQUIREMENTS ELICITATION

Once the system requirements are understood, the requirements engineer can then determine the software's function, its interfaces, its behavior, constraints, data elements,

and relation to the overall system. The process for gathering that type of data is called requirements elicitation. There are a number of methods for gathering data from users and developers.

Requirements can be broken into four overlapping categories. Business requirements represent the rationale for the system and its vision and scope. User requirements represent the goals of the user and the tasks that a user must perform. Functional requirements are the behavioral characteristics of the software. The last category is non-functional requirements such as quality factors, security, performance goals and constraints. (Heldman, Wiegers) The challenge is to capture all four categories of requirements from as many perspectives as possible.

Requirements elicitation can be a challenging task. The software engineer tries to develop precise, specific requirements from the stakeholder's (users, program managers, process owners) initial problem statement. In the beginning of the process, the stakeholders may not know exactly what he or she needs or wants. It is not unusual for stakeholders to make broad, vague statements, such as, the "system must be user friendly." It is also common that stakeholders present inconsistent requirements that are driven by a specific individual's wants, needs, or bias. Requirement elicitation involves intensive interaction with stakeholders to drill down into the problem to determine what the stakeholder wants the system to do. Once the requirements are identified, they can be analyzed and checked for such things as internal consistency and consistency with respect to policy and business rules.

This dissertation will use the term requirements engineer to describe the person responsible for the requirements engineering process. In many cases this individual is a software engineer and is part of the software-development effort, in other cases the individual specializes in only requirements gathering. This dissertation assumes that the requirements engineer is a contactor. This individual may or may not be involved in the development of the SLAs. In many cases outside contractors may be necessary to develop the SLAs to ensure objectivity, as most ESPs do not want SLAs. While requirements engineering and SLA development may be handled separately, good cooperation and information sharing is absolutely necessary to obtain maximum benefits.

The software-development organization may want membership in the SLA development team to ensure knowledge sharing is occurring. It is important to keep in mind that the quality requirements developed in the SLAs must be fed into the overall requirements engineering process.

Requirements elicitation requires a great deal of interpersonal skills. It is not always easy to get people to clearly articulate their ideas. Everyone has his or her own societal beliefs, biases, values, parochial interests, agendas, educational backgrounds, and perspectives. The requirements engineer needs to understand the beliefs of the stakeholders (epistemology), what is observable in the world (phenomenology) and what can be agreed upon as being objectively true (ontology). (Nuseibeh)

Given the different viewpoints, social and political issues, and the stakeholder's various perspectives, requirements elicitation utilizes a number of techniques to obtain complete, and accurate requirements. Traditional requirements elicitation methods involve meetings, interviews and group meetings with the various stakeholders to determine requirements. Some techniques include the use of collaborative software, group support systems and various scenarios or use cases. (Hickey) Others include ethnomethodological approaches (Sommerville, 1993), socio-technical modeling, stakeholder analysis methods, and artifact based elicitation. (Sutcliffe) In determining a methodology to use in requirements elicitation, the requirements engineer needs to understand an organization's perception of society and plan the approach accordingly. (Bickerton) All techniques have their advantages and disadvantages, and most software engineers utilize a combination of approaches.

One of the first steps in the requirements elicitation process to identify all of the stakeholders and external forces that provide inputs and or constraints to the system. Obviously everyone cannot be consulted for their input, so the requirement engineer must determine those stakeholders that can provide meaningful input. Then representatives from each of the major stakeholder groups need to be determined so they can be consulted, or included in the elicitation process.

Once stakeholders have been identified, the stakeholders need to determine the overall project or system mission or goal. Everyone needs to understand the problem that

the system is attempting to solve, as well as the means in which the intended system will solve that problem. The requirements engineer needs to gain consensus among all of the stakeholders on the problem to be solved and the approach that will be used to resolve that problem. "The primary measure of success of a software system is the degree to which it meets the purpose for which it was intended." (Nuseibeh) Requirements elicitation is concerned with discovering that purpose, and determining the functional, non-functional, and behavioral characteristics of a system that will meet that purpose.

The requirements that are generated as a result of interviews, market and environmental analysis, and interoperability constraints should be evaluated against the system's mission or goal to ensure that the requirements support and add value to the mission. Requirements that are 'must haves' need to be separated from those that are 'nice to have.' The task of determining the scope of the project becomes easier when the system and software engineers can tie the requirements back to the goals of the system.

Another important task during requirements elicitation is to document as much information as possible about each requirement. The documentation should describe the requirement, assign it with a unique identifier, list the stakeholders, classify the requirement by type, group the requirements into a parent/child relationship if necessary, and eventually assign a priority to the requirements.

As requirements are generated, the process of categorizing the requirements helps in the analysis process. There are numerous ways of categorization, and the methodology and detail used by the requirements engineer varies based on experience and the elicitation process being used. In many of the techniques utilized, the requirements are categorized according to whether the system requirement is part of the core system business process, whether it provides primary support to the process, or whether it provides secondary support. The core requirements describe the functional processes/actions that the system must perform to meet the system's mission. The primary supporting requirements are usually derived from the higher-level core requirements, from system constraints or interoperability requirements (other systems or data). The secondary support generally lists the quality or non-functional requirements, or requirements that are necessary to support the primary supporting requirements.

Requirements classification should also include whether the requirement is for the system itself (product) or for the process (standards, constraints, analysis model, etc…). Good requirements classification will help the requirements engineer assess the requirements to ensure that they support the system's goals.

Some common problems in requirements elicitation are managing the information from multiple sources (representing distinct viewpoints), tracing requirements back to their source and rationale, determining when the elicitation process has completed, and realizing that requirements are not always there to be elicited (there may not be a stakeholder). (Sommeville 1998, Sawyer)

SLAs assist requirements elicitation in four major areas. The development or modification of template SLAs provide an excellent starting point for group meetings, use cases based on those SLAs, or other techniques. The SLAs not only generate meaningful discussion, but they focus that discussion on non-functional attributes that are often overlooked. SLAs also tend to involve more management interaction in the requirements engineering process due to the contractual implications associated with SLAs. Template SLAs may address quality issues that the stakeholders did not consider.

This dissertation is making the assumption that the SLA development process is a part of the overall software requirements elicitation process. The SLA development effort can provide valuable feedback to the overall elicitation process. The team approach to the SLA development tries to ensure that all stakeholders are identified and that they are represented in the discussions. The SLA development/tailoring effort is usually a facilitated meeting with the stakeholders. The meetings allow brainstorming, debate, consensus, and can be a great way to identify conflicting requirements at the very beginning of the elicitation process. Group elicitation techniques aim to improve communication, foster stakeholder agreement and buy-in, while exploiting team dynamics to generate a richer understanding of needs.

In addition to identifying stakeholder requirements and needs, the formulation of SLAs helps the requirements engineer better understand the business domain, organizational culture, and operational environment. The process of developing SLAs is similar to the elicitation techniques of use cases and scenarios. Use cases describe the

interaction required between the users and the system necessary to meet the business objectives of the system. Use cases help to determine what users need to accomplish, as opposed to what they want the system to do or how it is expected to behave. The objective of the use case approach is to describe all of the tasks that users will need to perform with the system. (Weigers) It is important to keep in mind that use cases are from the perspective of the end user only, and should be used in conjunction with other requirement elicitation techniques to ensure all perspectives are accurately represented.

The process of developing the SLAs highlights and fosters discussion on the goals of the system, the processes and tasks that the system must perform to meet those goals, as well as identifying operational and organizational needs, policies, and constraints. Discussions necessary to develop the SLAs will generate information about the application domain, business and organization processes/culture, and the intended operating environment that the system will be placed in. The discussions will help the requirements engineer capture tacit knowledge, identify constraints, and justify how quality factors support business needs.

SLAs focus everyone's attention on quality factors at the beginning of the development cycle. Once the quality factors are included in the requirements, they will be incorporated into the design, and tracked throughout the product's lifecycle. The quality requirements will also be incorporated into the test plan at the beginning of development. SLAs also support the software quality metrics methodology recommended by IEEE. (IEEE Std. 1061-1998)

Quality factors can be affected by functional and non-functional requirements generated from sources other than the stakeholders. Requirements can also be generated from the operating environment, the application domain, regulatory or policy constraints, as well as interoperability constraints. Requirements can also be derived from the service needs of other systems in the environment. The formulation of the SLAs help to make some of these requirements explicit in that quality is affected by all of these requirements.

Additionally, SLAs are concerned with how quality factors or performance attributes will be measured. As such, requirements that cannot be measured,

requirements that do not provide value to the underlying business process, and requirements that are not realistic will not be proposed or accepted. The goal of requirements elicitation is to gather all of the requirements in a concise document with good objective outcomes that can be measured. (Heldman)

SLAs also focus attention on those requirements that directly support the system's goals. SLAs are difficult to write and they require time and resources. As a result, superfluous requirements and "gold plating" are less common. Additionally those requirements that do not directly contribute to the goals of the system are generally eliminated as they create additional work that cannot be justified to management.

In many organizations, managements involvement in and commitment to requirements engineering is low. As a result, requirements are not normally related to business visions and objectives. (Bubenko) SLAs help mitigate this problem to some extent, because the format of the SLAs requires that the development group tie the quality requirements to the business plan. The contractual penalties and incentives involved with SLAs tend to capture more managerial attention than requirements gathering alone. If an organization is not willing to devote the time and effort necessary to tailor or develop SLAs, it is a good indicator to the requirements engineer that management will probably not be devoting as much resource support as is desired.

Use of a template SLAs can be useful when management or market forces do not allow sufficient time for requirements engineering. It is much easier to take the template and modify the SLAs than it is to develop the SLAs from scratch. Instead of spending time on determining a SLA format, writing the SLAs, and deciding how to measure a quality factor, effort can be spend on determining which SLAs best support a particular business process, and determining the specific quality thresholds to utilize. The template SLAs are also helpful to illustrate to stakeholders and management what SLAs are, how they are developed, how they are utilized, and how they support business needs.

The SLAs in appendix (A) are an example of how SLAs can assist. Appendix (A) contains services and accompanying SLAs for hosting services. The SLAs are based on industry standards, common practices, and thresholds that the author felt were essential to provide support for the hosted applications. SLAs in appendix (A) are intended to be

used as a template. These template SLAs provide good building blocks to guide discussion and focus thought on non-functional requirements and quality; and they are intended to be modified to suit the system being developed. The fact that services and SLAs are already defined greatly assists both users and program managers in establishing their requirements with respect to hosting services. Instead of every program developing their own services and SLA requirements from scratch, they can utilize the template, which already represents good business practices. The SLAs in the template can be continually updated to reflect better/best business practices and lessons learned from other program's contracting efforts. The SLAs in Appendix (A) are specific to hosting services, but SLAs and the template concept can be applied to any stage of software development or lifecycle management.

When the requirements engineer feel comfortable that requirements from the representative stakeholder groups have been identified, the process of analyzing the requirements begins. There is conflicting guidance as to whether the process of validating requirements is part of the elicitation process or the analysis process, but in this dissertation the process of determining which requirements support the system's goals will be in the analysis section.

C. REQUIREMENTS ANALYSIS

The goal of requirements analysis is to identify the goals of the system and develop an acceptable set of requirements that will meet those needs. The requirements should be necessary and sufficient; there should be nothing left out, and nothing superfluous added. (Sawyer) The requirements engineer wants to ensure that the analysis verifies the goals and scope of the system, identifies the requirements necessary to meet the goals of the system, ensures there is sufficient documentation to evaluate the requirement, resolves conflicting requirements, assesses risk, identifies constraints, and assigns requirement functionality to the various software components.

This dissertation is also assuming that SLAs will be analyzed in the same manner as the requirements captured in the elicitation process. The requirements engineer will want to ensure the SLA thresholds support other functional requirements, that they are

technical and fiscally feasible, that the functional and non-functional requirements do not conflict, and that the SLAs support the overall system goals and quality requirements.

It is important to note that the requirements analysis phase overlaps with the beginning of the software-development phase. Software developers will start to become more involved towards the latter stages of the requirements analysis when requirements are allocated to software components, models and/or prototypes are developed, and the software architecture development is started.

Before the analysis process starts, the team that will conduct the analysis of the requirements should be selected. The team is generally composed of the requirements engineer, designer representatives, representatives from the stakeholders, and in some cases system engineering representatives will want to be on the team to ensure better integration between hardware and software. It is important that the users participate in the analysis process, as this will influence user acceptance and help to establish user expectations.

One of their first tasks of the analysis process is to review the documentation gathered on the requirements collected in the elicitation phase. Requirements should be as descriptive as possible to eliminate ambiguity and allow those analyzing the requirements the opportunity to assess the requirements in terms of support to the underlying business process, whether the requirement is excessive, or not sufficient enough to meet its goals, whether it is technically feasible, and how it will be verified.

If additional information is needed, the stakeholders will be asked to comment on the missing information. The requirements engineering process is not a linear-sequential model, various parts of the process acts concurrently, and often feedback or additional information will be needed from a prior phase.

Once a baseline level of information is collected on each requirement, the requirements engineer can start the process of analyzing the requirements. The requirements engineer needs to understand the domain environment, mission needs, underlying business processes, and the organizational culture in order to analyze the requirements. Once the requisite information is obtained, the requirements engineer must evaluate the requirements in the context of the stakeholder needs, business needs, and

environmental conditions in order to determine the requirement's implications, conflicts, interaction, scope, and feasibility. Part of the assessment is to ensure that the requirements either directly or indirectly support the system's goals. The requirements are also evaluated for costs, risks, organizational acceptance, and whether they can be verified. Requirements analysis must not only address functionality, it must also take into account non-functional attributes such as quality factors, as well as programmatic constraints (budget and schedule).

It is not uncommon for stakeholders to have conflicting requirements, differing solutions to the problem being solved, or to demand differing levels of quality. The requirements engineer is responsible for negotiating a resolution these conflicts. It is important that the solution be worked with the various stakeholders so as not to alienate one party for the sake of another. This is not an easy task as there are multiple political and social agendas at work. An important step in the requirements analysis process is to document the priority of the requirements. This is essential in order to analyze the priorities against the budget, and it may help alleviate some of the conflicts, or at least identify them up front. In some cases, the primary customers (those in charge of the most business critical processes) will have to make the final decision. In other cases, if a consensus cannot be reached, it will be necessary to have management dictate a solution in decisions that affect multiple stakeholders. Boehm's win-win model (In) was developed in an effort to resolve this type of requirement conflict.

The requirements engineer also needs to perform a risk assessment of the requirements in context of defined constraints on the system and development effort. The budget and time constraints may have a dramatic impact on the system development. Additionally, environmental, technical, interface, and implementation considerations also affect the development effort. Requirements also need to be evaluated against the project timetable to determine if certain requirements with new or complex technology will present a risk to the project schedule.

Another step in the requirements analysis process is to determine the scope of the system and how it interacts with its organizational and operational environment. This is best shown through the use of conceptual models. There are numerous models that can

be used to model the problem and proposed solution set. These models include OOA, formal models, data and control flows, state models, event traces, state transition diagrams, entity-relationship diagrams and user interactions. (Sawyer) There is no best type of model as the use of a particular type of model depends upon the skill set of the software engineering and design team, the type of problem to be solved, availability of certain tools, interface requirements, and user/customer input.

Once the requirements have been analyzed and accepted by the stakeholders, the process of requirements allocation, or assigning/partitioning the requirements to the various subsystems, software components and sub-components can begin. This process is part of the architectural design. The functionality assigned to the components and their interaction ultimately determines the extent to which the system will exhibit the desired properties. (Sawyer)

It is important to note that many of the non-functional quality attributes can only be satisfied by more than one component. The various components of the system must act together or interoperate with other components to achieve the specified quality requirements. Two examples are reliability, which depends upon the mean time to failure for each of the components involved in a particular function, and built in redundancy. Another is response time, which depends upon the speeds at which the servers, application, firewalls, and supporting LAN/BAN/WAN operates.

Once requirements have been allocated to components it is often necessary to conduct further analysis to start the process of translating the high level requirements into technical specifications that the programmers can use to code. Details of the application domain, interfaces, protocols, types of data to be used, legacy components, and additional technical requirements will be derived from each system requirement. It is often necessary to begin another round of requirements analysis as new questions are raised.

Some of the issues associated with capacity management should be addressed in the requirements analysis portion of software design. Issues as simple as the anticipated number of users can have a large effect on resource requirements. Servers will have

excess capacity if the estimated users are too small. On the other hand if the users are underestimated, the server and application may not be capable of handling that amount of concurrent users.

The process of developing SLAs helps the requirements engineer in the analysis phase as well. SLAs assist in the collection of documentation, risk analysis, and conflict resolution.

A dynamic and rapidly changing business environment has forced end-users to demand more and better services from IT service providers. Businesses are demanding faster speed, more flexible systems, and near real-time computing. SLAs are an essential part of measuring and making explicit those needs. Part of requirements analysis is determining both functional needs as well as user expectations.

SLAs help the analysis process by ensuring that the quality factors and requirements are quantifiable and verifiable. The process of generating the SLAs focuses the effort on determining quantifiable attributes of the quality factors being considered. In some cases proxy attributes that are quantifiable will have to be used. The SLA process specifies the metrics that will be used to verify if the requirement is satisfied, the method of measurement, and the acceptable threshold value.

Requirements must be verifiable; otherwise they are just wishes that can consume and inordinate amount of time and resources. In order to verify a requirement, quantifiable attributes must be assigned to the requirement. Quality attributes can be difficult to express in quantifiable terms, however, this is a necessity to determine if the desired quality attribute was ever attained, or can be attained.

SLAs assist with the determination of technical feasibility and risk assessment by forcing the stakeholders to quantify requirements and then determine how they will be measured. The specified thresholds also assist in determining technical feasibility. During the SLA development, each requirement should be justified by a business case, which defines the benefits that the requirement provides to the overall business plan. The quantifiable requirements allow the developers and designers the opportunity to assess whether the system can be designed to meet those requirements given schedule and fiscal constraints.

SLAs also help in assessing the risks associated with the requirements included in the SLAs. The software engineer and developers can assess the various thresholds specified in the SLAs and determine whether they are technically feasible with respect to other requirements, proposed architectures, external forces outside of the system, whether the organization has end-to-end control, whether the thresholds are realistic, and to what extent the threshold levels indicated in the SLAs will support the system's goals.

In some cases SLA development will address and solve some of problems associated with conflicting requirements before the requirements engineer has to intercede. SLAs should always be tied back to the business process that the system is supporting. A business case should be made for each SLA, and a prioritization of the SLAs should occur during the development process. The SLA development process should begin by discussing and agreeing on the system's goals, goal hierarchies and priorities, and project scope. Requirements supporting a particular stakeholder's agenda (social, political, or organizational) will not be supported unless that stakeholder can make the case that the requirement supports the system's goals and is worth the investment (SLAs will drive up the cost of the contract as it imposes additional risk on the developer).

The group effort to develop the SLAs also helps stakeholders understand other stakeholder's perceptions, politics, and desires. A group decision forces everyone to justify their requirements in terms all other stakeholders understand, disputes can then be based more on logic than emotion.

D. REQUIREMENTS SPECIFICATION

Requirements specification is the process of documenting the requirements. This generally takes the form of three documents. The first document defines the system vision and scope. This document is known by many names such as user requirements document, concept of operations, or scope and vision document, but it typically includes four parts. (Wiegers, Sawyers)

1. Vision and Scope Document

The first part of the vision and scope document outlines the business rationale for the new system. This section lists the reasons the system is being developed and its intended benefits. The intended benefits or business objectives of the system need to be defined in quantifiable terms so the success of the system can be measured. This section provides information on the customer base, target environment, and the risks associated with the project.

The second part is the vision statement. This section details the long-term purpose for the system or product. It describes why the system is needed, its intended customers, and how it is different/better than other systems already in the market place. This section also details the major functionality or features of the system as well as describing dependencies and constraints.

The third section describes the scope of the project. This includes a summary of the features that will be included as well as those that will not be included. This section is intended to focus the development effort and establish user expectations. Without a good scope document, there is a good chance of requirements creep. This document will also discuss release strategies. In some cases certain functionality is needed before others, so a base release with the main functionality may be planned, followed by upgrades introducing additional functionality.

The last section describes management issues associated with the project. This section lists the stakeholders, system functionality that concerns them, how they will benefit from the system, and their concerns. Project priorities are also outlined in terms of cost, schedule, features, and quality. Additionally this section describes the operating environment and the non-functional quality factors that will be necessary to achieve the business objectives. The document may also include a conceptual model that further illustrates the boundaries of the system, interfaces, data flows, and control.

The scope and vision document outlines the business case for the system allowing the requirements engineer to tie requirements back to the original business case. This is an important part of requirements tracing. As requirements change it is important to constantly evaluate them against the original business case to ensure that the

requirements support the vision and scope. If they do not, it may be necessary to update the vision and scope or disregard the requirement modification request.

2. Business Rules

The next document describes the business rules that apply to the system. The business rules incorporate internal business policies and procedures, regulations, formulas or algorithms that will be incorporated/impact the system, and external forces/market conditions that will influence or constrain the system. Generally each business rule has a unique identifier, a description, taxonomy or classification, and the reference.

3. Software Requirements Specification

The third document is the software requirements specification (SRS). The SRS states the functional and non-functional requirements of the system. This establishes the contractual basis of the agreement between the customer and the developer of the system. Since the SRS provides the foundation for project management, requirements verification, test and evaluation, design, cost estimation, and development it is essential that it describe as accurately as possible the behavior of the system under expected conditions. (Weigers)

There are several recommended standards for developing a SRS, including IEEE p123/D3 guide, IEEE std. 1233, IEEE std. 830-1998, ISO/IEC 12119-1994 and IEEE std. 1362-1998. (Sawyer). IEEE std. 830-1998 suggests a template composed of six sections and three appendixes. The first section discusses the purpose and scope of the system (readers can be referred to the vision and scope document if applicable), document conventions, and references. The next section is a high-level overview of the system, its intended operating environment, constraints, assumptions, and dependencies. The third section lists in detail the system features, the priorities attached to those features, and all requirements that are associated with that feature. The fourth section lists all external interface requirements, including user interfaces. The fifth section lists all non-functional requirements such as performance criteria, quality factors, safety, and security. The sixth

section lists requirements that were not listed elsewhere. The appendixes of the SRS included a glossary, any of the analysis models used, and a list of all outstanding issues. (Weigers)

SLAs can assist in the formulation of the specifications in three important areas. If a format similar to that used in appendix (A) is utilized, the sections discussing why the measurements are necessary, scope of the measurements, assumptions, and responsibilities, provides good information to incorporate into the specifications. During formulation of the SLAs, the format of the SLAs will drive communication and discussion among the stakeholders with respect to the scope of the system, ensuring that the requirements levied are necessary and support the business objective of the system, and that they are quantifiable and measurable.

The SLAs must be written to withstand legal scrutiny. The SLAs must be verifiable, concise, unambiguous, and understandable. The requirements that are included in the SRS should contain those same attributes. The examples included in a template SLA provide a good starting point that stakeholders can use for generating other requirements. If new SLAs need to be generated, the team will quickly discover the difficulty of writing clear requirements that can withstand the rigor of analysis by other team members, project managers, legal staff, developers, and/or the contractor. Either way, the lessons learned during the SLA development effort can be applied to the other requirements outlined in the SRS.

The SLAs can be written to ensure quality in the specification documents themselves. SLAs can specify quality factors such as adherence to specified formats, text structure, requirements labeling and tracing, completeness of the requirements (not all requirements must have all the needed information, but those that do not should be annotated, and tracked for resolution), and a consistent level of detail in the requirements. Davis has outline 24 quality factors that he feels are essential in a SRS. (Davis)

E. REQUIREMENTS VALIDATION

Once the specifications are written they need to be formally reviewed to ensure they are accurately represent stakeholder's requirements, and that the specifications

reflect the desired quality. The intent of the verification is to find any errors before they are incorporated into the design as the costs, as the cost to correct defects once incorporated into design is approximately 100 times more that it costs to correct them in the requirement engineering phase. (Cross) Poor requirements engineering will result in poor product quality, cost and schedule overruns, and poor customer satisfaction.

The validation is generally a formal inspection consisting of a team comprised of the software engineer's staff, stakeholder representatives, and developers. The group is looking for errors, omissions, assumptions, as well as implicit constraints and assumptions. Stringent quality factors may generate implicit validation requirements. For example, the requirement for an exceptionally high degree of reliability or safety may implicitly drive the need for formal specifications and analysis to determine if the quality requirements can be met. (Sawyer)

The verification of the quality of the requirements documentation is an essential part of the requirements validation process. The documentation should be validated to ensure that it conforms to established standards, is understandable, modifiable, consistent, traceable and complete. Validation also ensures that proper documentation or knowledge management is applied to the models. A documented history of the rationale, reasons, and trade-off discussions is extremely valuable as the project progresses and tacit knowledge is lost.

The requirement documents are also checked to ensure they contain the requisite amount of information necessary to validate requirement feasibility and necessity. The documents are reviewed to ensure that the requirements do not contain any significant conflicts, the specifications contain enough information to start design (concise with no ambiguity), the requirements are within the project scope, the software requirements support/do not conflict with system requirements, and that business rules were correctly applied.

Requirements validation also refers to the models that are used in the analysis phase of the requirements engineering process. Any modeling technique biases the perceptions or views of the stakeholder as they offer only a limited number of primitive concepts for modeling its intended subject matter. (Mylopoulous) Validation, through

formal walkthroughs or inspections will help to identify errors, emissions, or identify assumptions. Validation can also help to ensure that the modeling used is sufficiently robust to capture the problem and proposed solution.

The models also need to be validated to determine whether the analysis models accurately reflect stakeholder's requirements. One problem encountered when gathering requirements is the fact that a requirement engineer's perception and description of problems can be influenced by the tools and methods that they utilize to capture requirements. If stakeholders do not agree with that perception or frame of reference, then they are not likely to agree on the representation of the requirements.

The approach utilized to capture requirements can be broken into two philosophies. The first is a positivist approach where the requirements are founded and verified by empirical observation. This approach has been criticized because it tends to force stakeholders to model reality into neat empirical terms, where others argue that reality is not that simple. The other approach is an ethnomethodological approach that stresses value-free observations by not imposing modeling constructs. However, the synthesis of the information gathered must still be presented and communicated in some form. It is possible for the requirements engineer to taint the requirements with their own biases and social values. (Nuseibeh) The validation process checks the requirements to ensure the stakeholders, developers and management agree on the frame of reference and the resultant models.

Prototyping can also be used to validate both the models, and requirements. Prototypes are advantageous in that they can quickly demonstrate the requirement engineer's assumptions and allow stakeholders to provide feedback. However, prototypes can distract users from the core functionality by shifting attention to cosmetic user interface issues and any problems that may arise with the prototype. (Sawyer)

The validation process is also another check to ensure that requirements do not conflict. The conflict can be caused by numerous reasons, including problems describing the requirements in the specifications, new knowledge, problems missed in the analysis phase, new requirements as a result of prototypes or changing environments, missing requirements, and any aforementioned bias interjected during the requirements

engineering process. Disputes are not unusual given the diverse backgrounds, cultural differences, inter-organizational politics, and different approaches to solving the perceived problem. Disputes can be resolved through goal hierarchies, prioritizing requirements, utilizing Boehm's win-win model (In), and negotiating compromises.

SLAs can also be used to specify quality factors for the specifications and the models. SLAs can specify specific procedures and processes to utilize and it can specify the accuracy of the documentation. A designer or developer of the system can then know the level of quality that is contained in the specifications and models. Otherwise they would have to evaluate the models to ensure accurate notions such as events, states, cause and effect relationships, compatibility, and mutual exclusion.

A common problem found in software projects is that they did not quantify the benefits or risks of different designs and requirements. In many cases intangible benefits are not mentioned. (Bubenko) The SLAs are quantifiable which allows the requirements to be measured to determine how well a design solution satisfies the requirement. One of the most important traits of a requirement is that it is verifiable. If there is no way to determine whether the requirement has been met, it should not be included in the specifications. Part of the requirements verification effort is to determine whether a requirement as it is specified in the SRS can be verified, so an acceptance test can be developed to determine whether the system meets the requirement.

Template SLAs are also helpful in that many of the methods used to measure the non-functional or quality aspects of a system are already defined, and used in other projects. Verification of the SLAs is generally quicker that other requirements in that template SLAs have already withstood the scrutiny of verification. However, that does not mean that SLAs should not be verified, they must be reviewed in relation to the systems they support.

F. REQUIREMENTS MANAGEMENT

Requirements management is the process of documenting new requirements and ensuring that any changes to the system that affect the requirements or their supporting information are accurately documented. System and software requirements are not static;

they are constantly evolving as users gain more knowledge by analyzing models or prototypes, as designers discover omissions or need additional clarification, and as business environments are changed. As change occurs, it is important that the documentation on any requirements affected by that change be updated to reflect any new information. Requirements management occurs throughout a product's lifecycle.

Requirements management can be broken into three separate but related tasks. The first task is updating the requirement's documentation to reflect change. The second task is requirement's tracing, which is concerned with identifying sources and rational for a requirement, as well as identifying where that requirement is reflected in the architecture. The third function is an impact analysis of the proposed change.

Requirements engineers have to be very organized to ensure that all of the information on a requirement is captured accurately. The quality of the documentation is essential to the development effort as well as the life-cycle support of the system. Every requirement needs to have a unique identifier, a classification, dependencies on other requirements should be noted, hierarchical relationships and the requirement's rationale (why the requirement is justified and how it supports the business process) needs to be recorded, as well as the source of the requirements and the software component(s) it was assigned to.

Since change is inevitable it is necessary to have a management system in place to ensure that when the system or requirements are changed, that there is a mechanism in place to capture that information. The requirements engineer also needs to ensure that when change does occur that the documentation is updated. It is important to note that the documentation does not just include the specification in the SRS, it also includes all ancillary information that is used to interpret and manage that requirement. (Sawyer) In addition any context models that were used should also be updated to reflect that change.

In many cases system characteristics and user perceptions of need change faster than the requirements engineering process. (Bubenko) Changes in the design stage still need to be documented by updating the original specifications. When updating the documentation, it is extremely important to utilize version control of the individual requirements, as well as the vision and mission documents, SRS, and context models.

(Weigers) Automated tools are making the process of updating documents and version control easier, but multiple data formats (including conceptual models), distributed working environments, and extremely large and complex systems still limit the effectiveness of these tools. (Bubenko)

Requirements tracing is concerned with establishing links with the requisition (request for the requirement), its source, its specification documentation, other requisitions that would be affected by any change, higher-level system requisitions, and the business plan/process it supports. In addition the requirements should also be traceable to the design element that satisfies it.

It is very important that a requirement be traceable back to its source and/or rationale (business objectives, business rules, system requirements, dependencies, etc…). If there are no links between the business plan and the specifications, then it becomes very difficult to determine the impact that a change in the business plan/process will have on the system. It is also difficult to determine the impact that a requirement change will have on business processes. In addition it makes risk management difficult when changes in the business environment cannot be evaluated in terms of which requisitions are affected.

The third function of requirements management is performing impact studies on the effect of any proposed change. Changing requirements need to be assessed to determine the affect of the change on the system's cost, schedule, and performance. As requirements are changed a cascading effect can occur in which dependent requirements are affected, conflicts can be introduced, the architecture can be affected, and performance and quality requirements can be impacted. When the impact analysis is completed the software engineer, program managers, and stakeholders will have to determine if the change is necessary.

Another consideration when conducting impact studies of proposed changes is user expectation. Management and stakeholders need to understand the effect that requirements will have on the costs and schedule associated with the program. In some cases change is needed and should be embraced, but in other cases the change is a result of requirements creep. Impact studies may also inform stakeholders and management of

proposed changes that they may not have been aware of. Stakeholders spend a great deal of effort and time during the elicitation and analysis of requirements, and they will not be pleased if the system is modified without them being informed.

Requirements management is often a neglected part of the requirements engineering process. It can be very time consuming, it is difficult to manage, it is not glamorous, and it is often neglected in the rush to market a product. Additionally, programmers are notoriously poor at documenting anything.

SLAs can be are extremely useful in the area of requirements management because it institutionalizes a change management review board that is responsible for impact analysis and change approval. SLAs are contractual documents that generally have incentives or penalties associated with levels of performance or quality goals. Any changes to the system that affects those specified quality or performance goals must be negotiated as part of the contract. For example the new requirement for a content screening program on the e-mail system may affect performance thresholds. This new requirement may necessitate a renegotiation of the SLA.

As stated earlier, SLAs can be written to apply to the quality of the requirements documentation. Audits of the system and the corresponding requirements documents will determine compliance with threshold levels (probably a percentage such as 98% accuracy).

Requirements management continues throughout the system's lifecycle. A common problem with documentation is that once the system is fielded, it is turned over to another team to manage in its operational phase. The more accurate the documentation, the easier the transition to the new team, and the system will be easier to maintain. Unfortunately, there is generally little incentive to keep the documentation up to date, or accurate. This is where SLAs enforce some rigger.

G. SUMMARY

The central theme of this dissertation is that SLAs can help program managers and software engineers produce higher quality software. One of the ways that SLAs help is that they focus attention on the non-functional requirements of a system. Specifically

the quality factors (including performance requirements) that users, program managers, and software engineers feel are essential in a system to support the underlying business process. They also help make explicit many of the quality factors that users may implicitly assume. SLAs also specify the quality metrics by which the software quality factors are measured. Measuring and monitoring quality allows an organization to determine whether quality requirement have been met. Measurements also support early detection and resolution of quality problems.

The process of developing SLAs improves the requirements engineering process by involving all stakeholders in discussions that result in a common understanding of the business factors that drive the need for certain quality factors. Ends users and program managers collaborate to determine quality factors and performance characteristics as well as functional requirements. Those quality factors are taken into consideration when the system is designed, and that design is verified. In addition to specifying quality characteristics SLAs can be used to specify and enforce standards and processes that also lead to quality software development. The quality thresholds incorporated into the SLAs will also be represented in the testing scenarios to ensure compliancy. SLAs not only assist in the requirements engineering, but they are one of the first steps towards establishing a quality control process.

VI. DESIGN

The intent of this chapter is to demonstrate how SLAs can influence software design to achieve a higher quality product. The section will discuss how specific quality factors can drive design, it will discuss quality metrics that can be incorporated into SLAs that are specific to the design phase, and it will discuss how SLAs can help in the development of the test plan. This chapter will not however provide an in depth discussion on how software is developed, as that is outside of the scope of this dissertation.

SLAs can be used to specify quality factors specifically related to the design process, but SLAs real contribution to generating quality software is in the fact that the quality factors that are addressed in the SLAs drive the design. When customer requirements have been collected and specified, design is the process that translates those requirements into a blueprint that programmers can use to build the product. The design can then be assessed for quality, as it is a representation of the final product. (Pressman) The design model can be reviewed to ensure that the quality factors were adequately addressed. In this way quality is designed in the beginning phases of the lifecycle. Waiting until the testing phase of development to determine whether quality factors have been met is too late. It is much easier and less expensive to achieve specified quality factors if they are addressed at the beginning of the application lifecycle. Discovering problems during testing requires significant time in evaluating the symptoms and working backwards to discover a root cause. (Cross)

The quality factors identified in the requirement specifications enables the application developers to employ the pertinent technologies and products, in order to achieve a design that meets the desired level of quality. (ITIL) "From a technical perspective, quality attributes drive significant architectural and design decisions." (Weigers) If developers know which of the characteristics are most critical to project success they can select the architecture, design, and programming approaches that best achieve the specified quality goals.

The quality factors specified in the SLAs have penalties or incentives associated with them, as a result, the development team will focus more attention on ensuring those attributes are incorporated into the design. Program managers and developers tend to concentrate more on functional requirements than non-functional requirements. This is especially true when the program is experiencing significant schedule and/or monetary pressure. The SLAs help to ensure non-functional requirements are not overlooked in the design process.

A. ARCHITECTURE ANALYSIS

Where the requirements define what a system is supposed to do, the design represents how the system will do it. The architecture of a software system models or defines the system in terms of the structure, behavior, organization of computational components, and interactions among those components. (Shaw, Pressman, Bass) Architecture also shows the correspondence between the system requirements and the elements of the constructed system, thus providing some rationale for the design decision.

Software architecture is a compilation of design models representing the various aspects of the software system at different levels of abstraction. Although there can be numerous levels of abstraction, depending upon how far the designers want to decompose the system, there are three general levels of abstraction. The first level represents topographical arrangement of components (a unit of computation or data storage) and connectors (an entity that facilitates communication). (Dias) This level maps system requirements with components and describes the interactions among the components. The next level involves design issues involving algorithms, data structures, primitive operators, primitive language operators, and threads of control. The bottom level consists of design issues involving memory maps, call stacks, and register allocations. (Shaw)

The architecture also represents multiple views or perspectives of the system depending upon the information to be modeled. These different architectural structures or models are interrelated and provide a holistic view of the system. Some common structures are module structure, logical structures, process structures, physical structures, uses structures, call structures, data flows, control flows and class structures. (Bass)

These structures can also be broken into architectural design, data design, interface design, and component design. (Pressman) They can also be broken into functional areas such as presentation services, information services, communications, interface design, transaction services, environmental services, and base services. (Goodyear).

The different types of structures guide design with their own sets of components, notations, analysis techniques, and issues. In addition each structure may have multiple levels of abstraction, which also have components, rules of composition, and rules of behavior. Each structure and level of abstraction provides a unique perspective and can be considered a separate software blueprint. (Bass) The structures are not necessarily independent, as they will often overlap. As such, each structure and the interface with other structures need to be evaluated in terms of the quality factors defined in the SLAs.

As more business essential processes and functionality rely on IT intensive systems, it is not realistic to expect that organizations will take a vendor's word that the system under development will meet all of their quality requirements. The organization should be able to independently evaluate the vendor's architectural decisions and design as early in the software-development cycle as possible to ensure their requirements are being addressed. (Clements)

Software architecture analysis is used to predict the quality of a product before it has been built. The analysis provides information that can be used for architectural trade-offs, risk analysis, and to ensure quality factors have been addressed. Architectural analysis cannot be utilized to obtain qualitative measures (precise estimates) of the effectiveness of a particular design on a certain quality attribute. (Dobrica) As a result, architecture analysis provides support for SLAs by ensuring the design addresses quality requirements, but caution should be used when utilizing analysis results as threshold measurements. Although there are a number of methods to analyze architectures (Hulse, Dobrica, Garlan, Clements, Land, Bass), further work is needed before these models can produce qualitative or quantitative quality measurements needed for incorporation into SLAs. (Dobrica) They can however provide an estimate of how well the design will satisfy a particular quality factor.

SLAs also help to ensure that once the software architecture has been analyzed and accepted the architecture is not modified during the code phase. Although an explicit software architecture is one of the most important software engineering artifacts to create, analyze, and maintain, it is difficult for developers to remain faithful to an intended architecture as design and implementation proceed. (Cross) SLA penalties help to focus management attention on satisfying mission essential quality factors.

B. SOFTWARE QUALITY FACTORS EFFECT ON DESIGN

This section is intended to illustrate how quality factors can influence design. The designer must choose an architecture that not only meets functional requirements, but it must also meet quality requirements. In making that decision the designer needs to ensure requirements are met, risk are evaluated, trade-offs studies are performed, alternative designs are evaluated, and potential quality conflicts are resolved. This section will briefly discuss some of the design considerations in meeting three quality requirements, but it is not intended to be a detailed study on design approaches and their effect on quality.

1. Maintainability

System maintainability is important to the availability of the system and lifecycle support. Although the costs of developing a system increase as maintainability is improved, the end result is improved product performance and lower life-cycle costs. (Markeset) Although it is difficult to quantify an overall measurement of system maintainability, proxy attributes and scenario-based measures can be utilized in SLAs. The attributes generally assess commonly accepted software engineering practices and processes. Specific scenario based measures, such as the time it takes to recover the system from a power failure, can also be utilized.

There are numerous design considerations that will affect maintainability. Many of these properties can be measured utilizing automated tools once the code is constructed, but the designer must consider these properties before programmers begin to code. The software needs to be designed with maintenance in mind. There are a couple

of key design considerations that will help a design meet maintainability quality requirements, including modularity, testability, documentation, and complexity.

Modularity is the decomposition of the system into specific components that satisfy assigned requirements. These components are developed as part of the software architecture process. Modules should be highly cohesive (perform only one task) with low coupling (simple interfaces between modules). Other module characteristics that must be considered when designing for maintainability are intra-module control complexity, intra-module data complexity, and inter-module connectivity. Intra-module control complexity is concerned with the flow of decisions within a module. (Callis) This quality factor can be measured by the number of decision statements and nesting levels within statements (function calls shall not be nested more than 2 levels deep (Weigers)). (Callis) Intra-module data complexity measures the average number of live variables per statement, the span of variables, and the number of operators and operands in the average statement. (Callis) Inter-module connectivity measures information flow, including the number of information flows into a module, the number of data structures from which information is received, the amount of data produced from a module, the number of data structures that use that data, and the complexity of the information flow (Kitchenham)

Maintainable software is also designed for testability. Testability and maintainability have many of the same proxy attributes, as many of the characteristics that would make a program testable would also make it more maintainable. Some of these characteristics include operability, observability, controllability, decomposability, simplicity, stability, and understandability. (Pressman) Some of the design considerations for meeting these proxy attributes would include adopting common coding standards, managing change volatility, ensuring field verification so incorrect input and output are easily identifiable, functional separation, internal instrumentation, and error handling.

As discussed previously, documentation is essential to good maintainability. Documentation needs to be well organized, accurate, accessible, and it must contain the appropriate level of detail necessary. Requirement changes and modification during design is normal. If those changes are not documented properly maintainability suffers.

Trouble shooting becomes more difficult if requirements are not mapped or recorded properly, models and the architecture were not updated, new interfaces are not recorded, and design rationale is not explained.

The development team needs to have established procedures to ensure that once a change has been approved that all necessary documentation has been updated. An audit of the processes used to implement an approved change can indicate whether the new requirement was properly documented, requirements models were updated, whether the change was properly approved and recorded, whether the change was communicated to others, and whether the architecture was updated. SLAs can help ensure documentation is accurate.

Programmers must also document their code, so it can be easily audited. The comment lines in the code capture the programmer's tacit knowledge, and allows others to understand the programmer's decision making rationale. Once coding starts, another method to improve maintainability is to specify in the SLA an acceptable ratio of comment lines.

Complexity is another measure of maintainability. The less complex a program, the easier it is to maintain. There are numerous metrics that can be used to measure complexity. Two common models are McCabe's cyclomatic complexity and Halstead's theory of software science. Both models can be utilized throughout the development process to ensure that the system in not overly complex. Specific design considerations to reduce complexity include reducing lines of code and keeping operators and independent paths as small as possible.

There are many other design considerations and models that impact and measure maintainability (Pearse, Basili). Although maintainability of a system is difficult to measure holistically, specific metrics can be utilized in SLAs to influence design considerations. The SLA development team in coordination with the developers can select the metrics or models that will be used to measure maintainability.

2. Security

There is a fundamental tension between designing for functionality and designing for security. There are several reasons for poor security in today's software, including

lack of training on defensive programming techniques, programmers relying on compilers to identify errors, and the demand for novelty means that much software development is on the 'bleeding edge' and is thus less reliable. (Gilliam) Another reason for poor security is the lack of a code analyzer that can parse through code, identifying common software vulnerabilities such as buffer overflows.

The intent of a security SLA is to ensure that software security is incorporated into the design effort at the beginning of design efforts. If designers concentrate all of their efforts on functionality and wait until testing discovers security vulnerabilities, the result will be schedule delays, less than optimal security, and greatly increased costs.

The security of the system needs to be evaluated from a number of perspectives. The application, operating system, network (including firewall), data bases, PC, and the physical security of the host environment need to be evaluated for security, and all contain security metrics that can be utilized in SLAs. An end-to-end SLA for security is difficult unless all parts of the system are managed and controlled by one entity, however pieces of the system can be analyzed and designed with security in mind.

It is difficult to measure security as a holistic measure, however there are specific security metrics that can be utilized in an SLA to influence design. One way an SLA can be utilized to address security concerns is to write the SLA such that an independent auditor will evaluate the security of the software design and in coordination with the software developers, they can develop a plan to correct deficiencies. The SLA can stipulate the time necessary to perform the security corrections, or the SLA can mandate a percentage of the problems that must be corrected by a given date.

Some of the most common security vulnerabilities include buffer overflows, script injections, changing environmental variables, numeric overflows, race conditions, information exposure, default settings, and programmer backdoors. (Gilliam) Designers must also consider security vulnerabilities resulting from interfacing with other programs or systems.

To combat these types of security problems, designers need to concentrate their efforts on four security requirements: identification and authentication, access control, audit, and system integrity. (Goodyear) Identification and authentication ensures that the

system can uniquely identify an entity in a transaction. Each entity must have a unique identifier, and there must be a way to bind the identifier to the entity. (Goodyear) Designers should utilize strong authentication where at least two authentication methods (what the user knows, what the user has, and what the user is) are used. An example of a strong authentication is a smart card along with a biometric verification.

Some examples of design considerations that address authentication include ensuring strong passwords (at least 8 characters that incorporate capitals, numbers, and special characters), establish an authentication period where the system times out after a period of inactivity, (Kabay) utilizing encryption protocols such as kerberos, ensuring passwords are strongly encrypted, and support for tokens or smart cards. Controls also need to be established if the application is accessed via a portal where a single log on is utilized for all applications on the portal.

Access controls determine what resources an entity can utilize. Access controls will determine whether an entity has been granted permission to access a program or a file. The access controls also determine the rights that the entity has with respect to the resource (i.e., the entity can only read the file and not modify, or the entity has full rights and can read, write, save, delete). Access controls are usually implemented by access control lists (ACLs) which specify the entity or a group that the entity is associated with, and the types of access that the entity has been granted with respect to specific files, systems, databases, application functions, or other resources. Another way to implement access control is through role based access control (RBAC), which associates a job function to a set of resources, then assigns an entity to a job function. (Goodyear) It is also important to track those people that have root authority, and to keep root access to a minimum.

Design considerations include ensuring essential files, operating system ports and files, database files, and application functions are restricted by access controls. This also includes the ability to copy files. The designer also needs to evaluate any interfaces with other systems to ensure that those programs are only given the access that they need.

Auditing is the process of monitoring the system to record who accessed a specific resource and when. Designers can ensure logs capture the resources that were

accessed, the identification of the entity, times, what functions were performed, and the success of those actions. The logs should contain enough detail to allow security personnel to reconstruct events in the case of a security breach. It should also be powerful enough to be used as an analytic tool for determining the root cause of poorly behaving systems. (Goodyear)

System integrity is the assurance that a system's implementation (or component) conforms to its design. Virus and worm attacks are probably the best example of system integrity attacks. Other examples are faulty parameters (setting that can be exploited), operational misuse, and data leakage. (Goodyear, Kabay) Designers need to keep system integrity in mind when designing the system. Identifying all points where the program receives input from users and other programs and implementing procedures to authenticate, restrict, and validate input parameters will help to improve a system's integrity. In addition data integrity can be protected utilizing programs such as Tripwire.

Security also includes the communication between the PC, servers, and database as well as network security. Encryption, intrusion detection software, restrictive firewall policies, and security policies (remote access, placement of web servers) should also be addressed in the system design.

Test personnel need to evaluate the project in all phases of its lifecycle to ensure that all security requirements were considered and incorporated. They should also incorporate security requirements into their test plan to ensure security is evaluated in the development and testing phases.

3. Performance

Performance is another quality factor that is often ignored in the design process. Unless performance requirements are explicitly stated, developers will concentrate on ensuring the design meet functional needs. Performance is often not considered until the testing phase, assuming it is incorporated into the test plan. Unfortunately, if you design poor performance into a system, correcting the problems can be extremely difficult, resulting in cost and schedule overruns. (Loosley)

Performance must be measured throughout the software's lifecycle. To manage performance, SLAs need to quantitatively define performance goals, so systems can be

designed to meet those objectives. Performance models can be utilized to verify that the design incorporates the specified goals and test plans can be developed to ensure the performance requirements have been met. Once the system has been fielded, the system must be monitored and tuned to ensure actual performance meets requirements.

Software performance engineering (SPE) is a method for constructing software systems to meet performance objectives. (C. Smith, 1996) It is designed to augment other software engineering processes. There are 10 fundamental activities of SPE including identify key business factors, specify performance objectives and priorities, evaluate design alternatives, summarize application workload mix, predict performance, monitor ongoing software performance, analyze observed performance data, verify performance expectations, tune application or system, and manage ongoing system performance. (Loosley) SLAs drive many of the steps in SPE.

Part of the SLA development effort is determining the performance qualities that are necessary to support critical business processes. The development effort also needs to identify key business factors that will affect the processing load placed on the system. Information processing needs depend on statistics like the number of customers, number of customer inquiries a day, peak hours, orders per hour, service hours, anticipated rate of growth, scheduled business events (monthly close-out), and use of remote sites. (Loosley)

Performance is dependent on a given workload; therefore an anticipated workload should be included in any SLA with specific performance targets. It is important to establish a baseline workload for the SLA, so that performance issues caused by excessive throughput that is outside the scope of the SLA may be identified. Most of the components in the IT infrastructure have limitations on the level to which they should be utilized. Beyond this level of utilization, the resource will be stressed and the performance of the application will be impaired. (ITIL) For example, if the SLA is based on an average usage of 1,000 employees, and the application is actually being used by 10,000 employees, the service provider may not be able to meet agreed upon SLAs. In this case the service provider should not be held accountable due to revised user numbers.

A system's performance can be described in terms of workload (instruction sets or transactions), response time (the time to process a single unit of work), throughput (a

measure of the amount of work that can be done in a certain amount of time), resource utilization (the level of use of a particular system component), and resource service time (latency and queuing time for resources). (Loosley) Some qualitative performance metrics that can be incorporated into a SLA include speed (processing time, retrieval time, response time), throughput (transactions per second), and timing (soft and hard real time demands). Strict performance requirements significantly affect software design strategies and hardware choices.

Once performance quality factors have been determined, the next step is to develop models to assess the performance qualities of proposed designs, and select a design that best meets the performance requirements. The performance of a system must be evaluated in terms of the structure of the software program (instruction length, data accesses, instruction mix), and characteristics of the target system (CPU speed, bus width, operating system, I/O characteristics, memory). The performance should also be analyzed at a number of different abstractions. There are numerous models that can be utilized to predict performance qualities. (C. Smith 1998, Menasce, Lazarescu) These models tend to focus on the essential processes of the system, resource usage and speed, and queuing theory. The models used depends on where in the lifecycle the model is being applied, the skill level of the design team, the size of the system being developed, the time, resources and funds available, and the level of abstraction being modeled.

The models are usually grouped into analytic models and simulation models. Analytic models utilize queuing theory and mathematical analysis to evaluate the impact of all processes on each resource, then computing the delays each process experiences waiting for service. Simulation involves running a simulated process through a software model of the system, which includes modeling each resource, models of the queue for each resource, models of processes within the resource, a model of the clock, and running a simulated process. (Loosely) Depending upon the size of the application, its architecture and its distributed nature, multiple models may be necessary. In a client/server architecture, it may be necessary to model message communication between

the client and the server, as well as model application procedures at the client and server side to capture the application logic and the pattern of access to the system resources. (Menasce)

The results of analytic or simulation models can be used to validate performance quality factors specified in SLAs. However, the models should be independently verified, and the quality factors should be rather general, (i.e., a specific procedure should process in less than 5 seconds) as the models are estimates and are not intended to be highly accurate (formal real-time models are an exception). As the system progresses in its lifecycle more accurate testing can be performed against actual code. Performance models are used more as a method of evaluating different designs than providing accurate quantitative values.

Modeling performance is not without difficulty. Estimations at the source level have problems taking into account compiler optimations such as loop optimizations, copying global variables into machine registries, dead-code elimination and constant propagation. (Lazarescu) It is also difficult to account for constructs using dynamic data structures, recursive procedures, and unbounded looping. (Suzuki) In addition, as the level of abstraction rises, the structure of the software becomes more difficult to take into account as it becomes further removed from the abstract representation. (Suzuki) Approaches for dealing with these problems include modeling a program in terms of a pre-calculated instruction code size and execution time, or where execution time is a function of the number of instructions and the MIPS rating of the target system. (Suzuki)

The intent of including performance quality factors in SLAs is to ensure performance is considered in the design of the system. There are numerous design alternatives that can improve performance at the system architecture level through to the software components. Some design considerations include load balancing (managing how processes are input into the server), thread architecture (taking advantage of parallelism and multiprocessor systems), balancing disk traffic (storing data on disks efficiently and strategically), locking strategies (identifying where locks are necessary, and when), resource management (identify resource intensive processes and potential

bottlenecks), and optimizing code for space as smaller code fits in fewer pages, leading to a smaller working set, fewer page faults, and it fits in fewer cache lines. (Reilly)

High performing systems also demand efficient use of memory (strategic use of cache). Modern processors are so much faster than RAM that they need at least two levels of memory cache. Memory cache consists of the fast L1 cache and the slower, but much larger L2 cache. A reference to L1 may cost 1 CPU cycle, L2 may cost 4-7 cycles while reference to main memory may cost 12-100 cycles. (Reilly) If data that is used together (temporal locality) is not stored together (spatial locality), it can lead to poor performance. Arrays have excellent spatial locality, while linked lists and pointer based data structures do not. Packing data into the same cache line usually helps performance, but not necessarily on multiprocessor systems, as cache sloshing (different processors updating the same cache line with their data) may be a problem. Caching must be done carefully. If the wrong data is cached, it is wasted memory. If too much is cached, less memory will be available for other operations. Not enough cache will result in wasted processor cycles, as the information missed in the cache will have to be retrieved. (Reilly)

To meet the performance quality factors specified in SLAs designers will have to increase their attention on performance issues such as memory allocations, cache lines, caching data, thread proliferation, locking strategies, resources available in the host environment, blocking calls, efficient algorithms, and resource utilization. Performance models will help the designers to analyze tradeoffs and independent evaluation can verify that a particular design will or will not come close to meeting performance thresholds in an actual system.

C. DEVELOPMENT QUALITY

This section will briefly discuss how SLAs can be used to influence project and process quality. Chapter 1 mentioned a number of project and process metrics and models. This section will discuss a few of the project and process metrics, and whether they can be incorporated into SLAs to help improve software quality. The metrics chosen to measure project and process quality will depend upon the size of the project, the skill of the developers and program managers, time to market, funds and resources

available, the return gained from the measurement effort, and the ability of the metric to accurately measure the quality objective.

1. Schedule

It is very important to choose the correct metric to measure a quality factor. For example, cost, schedule and function are the most important metrics to a program manager. However, cost and schedule may not be the best metrics to utilize in a SLA. There are numerous models that attempt to estimate cost and schedule (COCOMO II, Function Points), but these models are not accurate enough to utilize in a SLA. Another difficulty is that establishing a software project's true duration schedule can be one of the trickiest measurement tasks in the entire software domain. (C. Jones, 1995) Determining when a project starts and is truly complete is difficult and must be precisely defined in the contract. In addition the pressure to meet those thresholds may result in the developers skipping important development steps that will ultimately result in large maintenance costs later in the lifecycle. It is difficult to develop a contract that is so all encompassing that the developers will not be able to "cut corners." Cost and schedule are metrics that are best included in the development contract, but not in a SLA.

2. Process Quality

One use of SLAs is to ensure that processes and standards are being adhered to. There are numerous standards that can be incorporated into SLAs. The SLA will specify the standards that must be adhered to and it will define the method to verify compliance. A third party can easily be utilized to verify compliance. Incorporating standards in SLAs provides a number of benefits. Standards provide a common methodology that makes management easier as they provide the basis against which activities can be measured and evaluated. (Horch) Standards are also useful in that they generally reflect industry best practices. Standards can be applied to development, coding, naming conventions, documentation, user interfaces, interoperability, architecture, and operating procedures. However, just because standards exist does not mean that they will be utilized. Incorporating standards in an SLA ensures that developers are aware of the standards, and that the standards will be incorporated into the development effort.

Some standards include ISO/IEC 12207and IEEE 1074, which specify processes, activities and tasks for software acquisition, development, operation, and maintenance that should be accomplished throughout an application's lifecycle. NIST 4909 (Wallace) and IEEE/EIA 1498 provide standards on documentation. IEEE 1059 provides standards on testing, as does ISO 9126. IEEE, ANSI, ISO and the Electronic Industries Association (EIA) have numerous other standards that can be incorporated. Although standards are useful, the SLA development team needs to be careful when selecting the standards to utilize. Some standards are very general and are open to much interpretation, and others may not be applicable to the project being developed.

Development processes can also be specified in a SLA. Specifying specific processes has many of the same advantages of specifying standards. Applying well defined, standardized software-development processes increases software quality and makes the development effort more cost effective and predictable. (Gnatz) Specifying the processes in the SLA helps to ensure that they are recognized and adhered to. Unless processes are contractually mandated, cost and schedule pressures quickly become more important, and necessary procedures are skipped.

One example of a commonly utilized development process standard is the CMMI model. The CMMI model defines specific key performance indicators (KPI) that must be established to obtain a specific level. A SLA can easily state that a development agency must abide by CMMI level 3 or higher. The Software Engineering Institute can be used to validate compliance. Many of the KPIs cover procedures that need to be performed to ensure a quality product. However, it must be noted that just because an organization has a process in place, it does not mean that they are utilized on a specific project. The SLA needs to be specific that all procedures at a particular CMMI level are in fact applied to the project, and that they are applied correctly.

3. Defects

Another common metric used to measure the effectiveness of a development effort is the amounts of errors found at a particular milestone. Some common metrics include defect density per software product, defect density per lifecycle phase, defects found by review, defects found by testing, user detected defects, cost of defect detection,

cost of defect correction, requirement errors as a percentage of total errors, defects incorrectly corrected, mean time to correct a defect, trouble tickets outstanding, and anticipated defects based on statistical analysis. (Horch)

Incorporating defect rates in SLAs is intended to encourage developers to implement their own software quality control procedures. Most development plans will contain formal quality control procedures such as audits, code walkthroughs, and testing. These plans should detail the quality control procedures, when they will be applied, and by whom. The quality control procedures are intended to measure product quality and provide feedback on the development process. Any errors found during the reviews or tests can be corrected and analyzed to determine their cause. Unfortunately, there are some developers that rely almost entirely on testing to discover any defects. This approach will ultimately result in more maintenance and costs. SLAs can be utilized to ensure reviews and audits are performed by third party inspectors at significant milestones. SLAs can also be utilized to ensure that the errors identified in the reviews are corrected.

A common metric that can be utilized in a SLA is defect density per KLOC (no more than 6 defects per 1000 lines of code). When dealing with defects it may be a better strategy to offer an incentive rather than a penalty. The goal is to encourage the developers to do their own internal reviews before the formal reviews to ensure they are using proper standards, procedures, and quality control procedures to analyze and correct defects.

If defects are used in an SLA, it is important that all stakeholders, including any third party auditors understand the definition of a defect, what constitutes a significant defect and what does not, and the methods that will be used to audit the project or product. A defect can be defined in terms of documentation errors, code errors, standards violations, requirements that were not met, improper output, model errors, module attributes (cohesion, coupling, complexity), or scheduling errors.

The SLA should also establish thresholds based upon the severity of the defects discovered. Stakeholders need to determine the various categories of defects and rate them based upon their impact to the mission and quality of the system. All errors do not

need to be fixed immediately, as some errors will not affect the functions or performance of the system. Those errors should be identified and fixed at a later time, as more effort can be expended working on more significant problems.

Defect audits have the potential to anger or demoralize the development team. Nobody likes to have their work scrutinized by personnel outside of their organization. The fact that audits are designed to improve the overall quality of the product needs to be stressed. The program manager will have to work hard to ensure that everyone views the audits and reviews in a positive light. This is one reason to utilize incentives rather than the more negative connotations of a penalty. Another approach is to write the SLA such that a percentage (95%) of all identified defect must be accurately resolved based on results from a follow-up inspection.

D. TESTING

This section will demonstrate how SLA development can assist the test community in the development of their test strategy. The main goals of testing are to challenge the software implementation of the requirements and the early detection of problems. Testing needs to be performed throughout a system's lifecycle to predict and evaluate the quality of the proposed design and implementation. SLAs can assist the testing and evaluation process in a number of ways, including identifying business critical processes, defining quantitative metrics to measure quality factors, identifying testing procedures, and ensuring testing is conducted throughout the system's lifecycle.

Much like the development effort, testing must be carefully planned, designed, executed, and reported. The test strategy outlines how the software system will be tested throughout its lifecycle and at the end of each development phase. It specifies what will be tested, when it will be tested, how it will be tested, the type of test needed, who will perform the testing, who will witness or verify the testing, what resources are needed (hardware, software, tools), calibration requirements for equipment, and acceptance or exit criteria. Part of the SLA development process is determining how quality factors will be verified. The SLA development process facilitates communication between the developers and the testing community at the beginning of the development effort.

Developers and testers need to have procedures and processes in place to identify and remove errors during requirements engineering and design before they are translated into code. (GSAM) Developing the SLAs will encourage both communities to develop a mutually agreed upon test strategy for the quality factors. Hopefully, this communication will encourage the developers and testers to also collaborate on a test strategy to address the functional requirements.

The software-development plan should detail all of the processes to be performed at each phase in the lifecycle. Each process should have deliverables, which will be validated and verified. Verification ensures the deliverable is complete, correct, conforms to standards, and was developed using proper procedures. Validation checks that the deliverables satisfy specified requirements (requirements tracing), and ensures that the deliverable does not have unintended consequences. Once the deliverables have been validated and verified, testing will be conducted to ensure that each specification has been properly implemented or satisfied. (Goodyear) These phase end reviews include the software requirement review, the preliminary design review, the critical design review, test readiness review (against product baseline) and the formal acceptance audits. (Horch) SLA can be used to ensure that phase end audits are incorporated into the test strategy and that they are performed.

SLAs can also ensure that other audits are performed. Some other audits include documentation reviews, requirements reviews, design reviews, test plan reviews, user documentation reviews, and implementation reviews. (Horch) SLAs can also ensure that certain tests are performed to ensure quality factors are being addressed. Some of the tests include unit testing, module testing, integration testing, coexistence testing, system testing, user acceptance testing, performance testing (stress tests), implementation testing, regression testing, and pilot implementation testing. (Philcox)

The amount of time, effort, and money that needs to be devoted to the testing effort is often underestimated. It is not uncommon for standard systems to spend between 50 and 80 percent of the development budget on test related activities (test execution, analysis, and error resolution). It is impossible to fully test a program. (Kaner) Traditional testing approaches only cover approximately 40 percent of the application

code. (Goodyear) SLAs help the test effort by focusing attention on the business critical processes that were identified in the SLA development process. As a result testing can be prioritized and focused on those processes that present the greatest business risk.

SLAs specify quantifiable quality metrics. These metrics should be incorporated into the test plan to assess the system's quality. This helps to guide the testing strategy and it prevent situations where the test program is aimed at showing that the software, as produced, runs as it is written, instead of challenging requirement compliance. (Horch)

To the extent that SLA encourages testing and the involvement of developers and the test community, it also drives testability in the design. Several key drivers for testability include fault tolerance (log data errors rather than allowing a crash), controls (input validation, access control, database balancing), error handling (identify and log errors), multiple operating modes (the system should have a production and test mode), and self-testing (validation of entry criteria). (Goodyear)

E. SUMMARY

SLAs improve the quality of software by incorporating quality factors into the development effort. The product quality factors specified in the SLAs drive design in much the same way as functional requirements. SLAs force quality to be addressed at the beginning of development and SLAs ensure quality is monitored throughout development. Once quality requirements are identified, the developers can select an architecture and design a system to best meet those goals. The test strategy will measure and evaluate those quality factors throughout the lifecycle to identify any areas that may not meet quality requirements.

Process quality and development quality can also be addressed by SLAs to improve the overall quality of the software. Although adherence to standards and processes does not guarantee a quality product, their use will greatly improve the possibility of obtaining higher quality. Monitoring the quality factors associated with process and project quality will also help to quickly identify problem areas and risks so they can be addressed early in the lifecycle.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. SOFTWARE QUALITY FACTORS

This section on software quality factors provides additional information on how quality factors are determined. It is expected that the processes discussed in this section were performed during the SLA development and/or requirements elicitation. The intent of this section is to demonstrate some of the difficulties associated with determining which software quality factors to utilize, and how the template SLAs can provide some help in making that determination.

Determining software quality factors that contribute to the success of the system or project can be difficult. It is easy to state that a system must be maintainable, available, dependable, portable, usable, or secure, but determining the correct level of abstraction to apply those factors, and quantifying them is more difficult. This difficulty is one of the reasons that non-functional quality factors are not always incorporated into the requirement specifications.

There are numerous quality schemes. Chapter II outlined some of the models. Papers from Charette, McCall, Boehm, and ISO 9126 discuss quality factors and their applicability to various situations. However, a detailed discussion of quality factors and quality metrics is beyond the scope of this dissertation. Instead, the purpose of this section is to discuss a methodology for selecting quality factors, highlight some of the difficulties associated with some of the quality factors, and propose how template SLAs can assist in the selection of quality factors.

A. DETERMINING QUALITY FACTORS

Chapter I outlined four areas where quality factors can be applied. This section will illustrate an approach to determining product quality, although this approach and discussion has applicability to project, process, and post-production quality factors. IEEE standard 1061-1998 presents a good framework for determining what product quality factors are needed and what metrics will determine whether those goals have been achieved.

The first step is to determine the quality factors for the system. The quality factors specified for the system requirements also need to be incorporated into the software components of the system. In addition to system quality factors, the software will need quality factors to ensure the software supports the underlying business process. Each of these quality factors should have direct metrics that specify quantitative measurements. In some cases it will not be possible to directly measure a quality factor. It may be necessary to specify surrogate or proxy attributes during each of the development stages. For example, code complexity can be a surrogate for reliability, testability or verifiability. (Schneidewind 1997, Weigers)

Part of this step is to determine those qualities that contribute to project success. The quality attributes may be prioritized based upon criticality to achieving a project goal, or it may be based upon a return on investment. Regardless of the methodology used to prioritize the quality factors, the fact that they are prioritized makes conflict resolution easier. The requirements engineer and the stakeholders can then evaluate the alternative design options and determine a solution that will satisfy the requirements.

The next step is to assign quality sub-factors to the software quality factors. This is essentially decomposing the quality factors into measurable software attributes. Building goal trees can assist in finding sub-factors. An example is the quality factor 'usability' which may be further decomposed into flexibility and sharing of information. Flexibility may be further decomposed into future growth and flexible work processes. Future growth can be further decomposed into design for extra personnel and design for modularity. (Mylopoulos) The quality sub-factors are usually more tangible and have greater meaning to programmers and analysts.

This step also focuses on the object of the measurement. Different parts of the same project may require different quality factors. In a N-tiered architecture, the front-end piece may need the quality factor of 'usability', whereas the back-end database may need the quality factor 'security' or 'integrity'. Differentiate the quality attributes that apply to the whole system from those that apply to specific components. (Weigers)

The final step is determining the specific metrics to assign to the sub-factors. This phase will also assign threshold values to the metrics and identify the means to measure

the metrics. This decomposition of quality attributes or factors helps the requirement engineers and software architects better understand the application domain, as well as highlights potential conflicts between the software goals.

This process should be evaluated at each stage of the software's lifecycle, and as changes are made to requirements. It is important to note that measurements obtained early in the development lifecycle will not be as quantifiable as those in the later stages of development. As development progresses, requirements and processes will evolve; those artifacts measured during requirements analysis will generally not be the same as those measured in the testing phase. In the early stages measurements will be taken on static objects such as architecture design, or specifications. In the later stages the measurements will be taken on dynamic objects such as the code itself. (Schneidewind 2002)

Template SLAs are SLAs that have already been developed for specific services. Template SLAs represent the best of breed or industry standard. Although there is currently not an industry standard, appendix (A) represents an attempt at establishing a template SLA for host services. Template SLAs that can be used to help in the quality factor selection. In many cases the user and program manager do not know what quality factors to utilize, nor do they know how to prioritize the attributes. Questions such as how reliable does the system have to be can be difficult to quantify. In the elicitation and validation process, requirements engineers are able to use methods to extract this type of information from users, but template SLAs are a good place to start in that they provide good examples of the types of software factors and goals that other organizations felt were important to their projects. Template SLAs also provide good examples of the level of abstraction to apply specific quality factors as well as presenting a scenario that illustrates the number of software factors and thresholds that should be used. It is not unusual for organizations to collect too many measurements. Excessive information is difficult to manage, and often leads to casual analysis or frustration. (Baker) Finally, template SLAs help the program manager by defining the quality metrics, specifying thresholds, and identifying their method of measurement. Some quality factors can be

difficult to define. For example the quality factor of usability can only be used in the context of the target user population, but it is often developed from the program manager's perspective. (Nuseibeh)

B. CONFLICT RESOLUTION

"Excellent software products reflect an optimum balance of competing quality characteristics." (Weigers) Determining the optimal balance is difficult in that the users, program managers, and developers all have different perspectives, and their respective quality factors will be determined from that perspective. Each stakeholder will have different priorities supporting the qualities that they feel best meet their needs.

The requirements engineer must first collect all of the quality attributes that the stakeholders feel are important. The next step is to work with the stakeholders to prioritize the quality factors. The goal of prioritizing the quality attributes is to focus on those attributes that best support the mission or goal of the project. Prioritizing the quality factors is important because some quality factors conflict with one another. The prioritization helps in the resolution of any possible conflicts.

Resolving requirements conflict is not easy as some combinations of quality attributes conflict with one another. It is important to understand the interrelationships that exist between the various quality attributes. Some attributes complement each other such as reliability and availability or flexibility and portability. Other attributes do not work well together. The attributes of flexibility and security often conflict as the measures to make an application secure also make it less flexible.

Attributes, such as efficiency, conflict with numerous other quality attributes. Tight precise code often conflicts with maintainability, portability, interoperability and flexibility. Additionally, attributes such as flexibility, usability and portability often conflict with performance goals. (Weigers) It is important to understand the trade-offs associated with each quality attribute as the choice of attributes will drive the architecture, coding, and testing. Understanding the attribute trade-offs also helps to form or manage user expectations.

Template SLAs can assist in the determination of what quality factors were important to other projects. The template SLAs can be used as a case study to see how other organizations weighed the benefits of the various quality factors against the mission or goals of their project. The template SLAs can be used as a starting point to determine which attributes are important to the stakeholders. The requirements engineer can then work on prioritizing the attributes, weighing the trade-offs, and resolving conflicts.

C. RESPONSE TIME

This intent of the next two sections is to discuss two quality attributes in depth and illustrate how they can be incorporated into SLAs. The focus of these two sections will be on the post-production phase of the lifecycle. Many of the issues discussed in this section were debated and the end result was incorporated into the SLAs in appendix (A). This section will discuss response time as a quality attribute, and the next section will discuss availability.

Response rates are extremely difficult to measure, and may in some cases, be too difficult to utilize in SLAs. The quality metric response time is a good indicator of customer satisfaction. Many quality metrics are technical in nature, but response time maps well to end-user's needs. If an application does not respond within a certain time parameter, the user becomes frustrated and their perception may be that the IT department or the service provider is not doing their job, or that the application does not meet quality requirements. Response rates are most useful from the perspective of the end-user. When a user enters a command, that individual is only concerned with how fast an answer or response is provided. Therefore, an end-to-end measurement of response time best satisfies the end user.

Response time is generally described as a measure of how long it takes from the time a transaction is initiated until all of the results are received. However, this definition needs additional clarification for use in a SLA. The definition must state at what point measurements begin and when they terminate. Additionally the SLA must state how response time will be measured. The definition above assumes an end-to-end response time from the client to the server and back, but the service provider may not own the

entire infrastructure. Many organizations have included response time SLAs in their contracts, but most of these SLAs do not adequately define the parameters of the measurements, nor do they define how measurements will be conducted. The measuring of response times is a complex process even if the service is an in-house application running on an intranet. (ITIL) It is very important that the SLA defines response time in sufficient detail that all stakeholders understand its meaning and how it will be measured.

End-to-end response times are possible when working within an intranet structure, where the PC, server, and infrastructure are all owned and operated by one provider. Unfortunately, this architecture is rarely the case. In the case of the Navy/Marine Corps Intranet (NMCI), the PC is owned and operated by Electronic Data Systems (EDS), the infrastructure to the outbound firewall is owned by EDS, the NIPRNET connectivity for the DoD intranet is managed by the Defense Information System Agency (DISA), external connectivity to the Internet is either managed by DISA, or contracted with local service providers such as SMARTLINK (AT&T), an application's server and host environment may be owned and managed by another service provider, and finally the application itself can be run by a Navy activity, DoD, or a commercial service provider. In this scenario it is extremely difficult to guarantee any level of service, since no one provider owns all of the pieces between the PC and the application.

The distributed nature of today's environment further complicates response time SLAs. Applications may have to query back-end databases over the Internet to gain the information necessary to satisfy a request. In this case, Internet latency can significantly affect response time. Issues such as bandwidth and control over the database are also issues. If the same service provider did not manage all of the servers in the tiers it may be necessary to specify response times for the various tiers at the server level. For example, when a front-end application receives a HTTP request, it may be necessary to measure the time from receipt of the request until the web server sends a request to the mid-tier server.

It is possible to study the service level contracts that have been negotiated with each of the component service providers and develop an overarching response time. For example, in a scenario where there are three service providers covering services from the

PC to the firewall, Internet access, and a host service provider, the response time for each can be added to determine a threshold. If the service providers agree that 1-second is an acceptable response time for their portion of the transaction, then an end-to-end SLA can be written for 3 seconds. In this scenario, a separate SLA will have to be negotiated with each service provider, or if there is one overarching organization responsible for the compute environment, then the third party agreements with other service providers will be tallied to arrive at an overall figure.

In reality this scenario is still difficult to manage and enforce. The application may have to be reengineered to incorporate certain APIs, time stamps, or exceptions to gain the response time information or monitoring devices would have to be established along the route from the PC to the server and back. All of the monitoring devices must be synchronized to identify and track a specific transaction. This would require that the service providers allow agents or monitoring software to be installed within their portion of the transaction. This may pose too much of a management challenge and security risk for most ESPs, as they do not want every client insisting on installing their own monitoring devises.

One problem with specifying response times with an application is that certain application functions may take longer than others. Some financial applications can take hours to calculate end-of-month returns. The question is whether it is possible to identify specific transactions and track and record their response time. If this is the case, then the application owner will have to identify those functions that are business essential, determine a response time threshold, and then tie a response time SLA to the specific functions. For example a web server should load a page within 2 seconds, while a database may take 30 seconds to a minute to execute a complex report. It is best to survey users to determine what response time is adequate for a given transaction. Typically the minimum and average measurements of response time are of interest. Benchmark studies of similar types of transactions can help determine acceptable thresholds for different types of queries.

It is also difficult to measure and aggregate the response times for multiple threads within the same program. If a session on the server consists of numerous threads

that in turn produce additional threads, some of which may execute distributed or sequential tasks, can the agents or measuring devices aggregate the total output? This becomes even more difficult if processing occurs on both the servers and the PC. Especially if some of the events are sequential.

Determining the cause of a delay may be difficult. If the operating system (OS) is the cause of the delay, how is that information being captured? Network and firewall delays, Internet latency, application errors, and user errors can all contribute to slow response times. To effectively isolate the cause of delays, monitoring devises will have to be installed at the various pieces of the infrastructure.

Another difficulty in accurately measuring response time is that the software performing the monitoring must be able to identify inputs, and the corresponding outputs. This means that whatever software is performing the packet sniffing operation must be able to not only identify the header addresses, but it must also be capable of determining packet content and determining whether the packets are inputs to a transaction, or are simply communication protocols. They must also be capable of determining whether the application is responding to the input in sequential order. If the server receives input 1 and input 2, before responding, can the software determine if the server is responding to input 2 before input 1?

If an end-to-end measurement appears to be too difficult, another approach is to monitor response time on the server itself. This approach does have some drawbacks. From an end user perspective, this is not a satisfactory solution as the application response time is the only part that is measured. It is not representative of the end users needs. Coordination problems with tracking individual inputs and their associated outputs still occur in the server. Additionally, the overhead associated with recording response time for applications with hundreds of concurrent users may actually slow down response time.

Rather than attempt to monitor the response time for every session, it is much easier to utilize the windows consoles on the server to run a program on the server itself that will measure response times to specific inputs. This type of a program is essentially

a synthetic transaction. In essence this is an end-to-end measurement from the server console through the operating system, to the application, and back.

In this approach, the program manager needs to determine the most important application functions to monitor. A program can then be developed to send input representing the various functions to the server to monitor response time. The program can execute at various times, testing all of the functions, or selecting individual functions randomly. This approach measures response times using statistical analysis, and is not concerned with attempting to measure response times for each concurrent user.

This program could also be run remotely using active X, although this will not be allowed under NMCI, and will probably not be allowed through the server environment's firewall. To ensure that the service provider does not tamper with the results, the server can e-mail the results to both the program manager and the service provider. A read only file will not work as the service provider has root authority, and can change permissions.

One disadvantage with this approach is that a program has to be written to perform the synthetic transactions. A third party solution would be preferable, and some do exist for testing web sites, but application specific transactions will have to be developed. Benchmark tests can help determine response times for each function executed. The response times for specific synthetic transactions can be incorporated into a SLA. Although this may not satisfy the end-user, it will ensure the server is operating effectively, and it will help to trouble shoot problems.

If response time SLAs are used, automated tools are essential in measuring compliance with the threshold requirements. SLAs that require help desk calls to determine whether response times have or have not been met should be discouraged. Automated tools are a necessity to remove the subjectivity associated with determining if the service is responsive or not. Help desk metrics put all of the reporting responsibility on the end users and the help desk approach also does not scale well. How many people have to report the incident before it is considered a violation of the SLA? What if there are thousands of potential users?

The SLAs in Appendix (A) do not contain response time as a quality metric. It was too difficult to develop a SLA given thousands of different applications, multiple

service providers, and security concerns. Synthetic transactions can be used, but each program will have to determine whether they want to use that approach or not.

D. AVAILABILITY

Availability can be defined as the ability of an IT service or component to perform its required function at a stated instant or over a stated period of time. (ITIL) Availability indicates the percentage of time that a system or service is expected to operate satisfactorily. (Wang) The formula for computing availability is composed of reliability and maintainability data. Reliability is the probability that a system will not fail. Reliability is generally defined in terms of the mean time between failures (MTBF) or mean time to failure (MTTF). Maintainability is defined as the time it takes to repair the system and restore it to operating condition. Maintainability is often expressed as a mean time to repair (MTTR). A common formula for availability (a) is a = MTBF/MTBF + MTTR. Another formula is uptime/uptime + downtime, where uptime consists of operating time and standby time, and downtime consists of unscheduled and scheduled downtime. (Hurst) Although the formulas appear to be straightforward, availability is difficult to incorporate into a SLA.

Overall availability is a function of the availability of the components (hardware, network, application software), the speed at which failures can be identified and repaired, the skill sets of the support personnel, the complexity of the infrastructure and application, the security of the system, logistical support, built in redundancy, and the application of tested procedures and processes.

Availability directly influences business and user satisfaction. However, unlike response time, availability is more technical in nature and does not map as well to an end-user. Many argue that response time is a better indicator of customer satisfaction. Some even argue that an availability quality metric is not necessary, as problems with availability will be reflected in response time measurements. For example, a server may be available, but the application may not be usable due to delays as a result of too many concurrent users. Response times would indicate situations where on-line shoppers disconnected due to slow processing times, where availability may not.

There has been much discussion on whether SLAs should concentrate on the technical side which concentrates on metrics associated with server, OS, infrastructure and application performance, or should be SLAs really be concerned with the perceptions of the end-user and the business processes owners. If the end-user inputs a transaction and receives a result within an acceptable time, are any other SLAs really needed? Is it necessary to specify server performance thresholds (CPU utilization, available table space) if the application is responding to input requests within specified time frames?

The author believes that, if possible, both response time and availability should be included in SLAs. Availability metrics require that the network, servers, and operating system be monitored for performance compliancy. This monitoring activity is essential in performing trend analysis, capacity management, troubleshooting, and measuring the effects of configuration changes. Availability monitoring is a proactive measure that will help to alleviate problems before they occur. Response time monitoring is reactive in that it will only report a problem once it has occurred.

Before SLAs can be determined for availability it is necessary to determine the level of availability that is needed by the application. Availability thresholds must be realistic. The higher the availability needed, the more costs will be incurred. If a system has an availability of 99.9 percent, the cost of improving the system's availability to 99.99 increases from 5 to 10 times for every additional 9. (Factor) A cost benefit analysis is highly recommended to determine the business losses or opportunity losses resulting from application downtime as compared to the price of maintaining a certain level of availability.

Availability is another area that is difficult to manage if the entire supporting infrastructure is not owned by a single entity. Unless the contractor has control over the PC, the entire infrastructure and the server, end-to-end SLAs will be difficult. Before any end-to-end agreements are made, the program manager needs to review the proposed SLAs with the service provider and all other third party service providers. It may be necessary to review the agreements with each infrastructure service provider to ensure

that the appropriate conditions and controls necessary to comply with the SLA are met. However, this assumes the contractors and third party providers are willing and capable of meeting proposed SLAs.

To properly determine an end-to-end SLA for availability, it is necessary to map and monitor all of the components necessary to provide full functionality. The reliability of each component must then be determined. Components can include server and network hardware, operating system software, as well as application software. It is important to remember that in order to achieve an aggregate reliability figure for a system, the reliability of each component is multiplied. If three items (PC, network, server) have 99 percent reliability, their aggregate reliability figure is $.99^3$ or 97.03 percent. The reliability of all of the component pieces in the system will determine the end-to-end SLA.

If reliability is the probability that a system will not fail, then it is essential that the SLA define what a failure consists of. That definition will also drive how the application, server, and infrastructure are measured and monitored. Is a failure defined in terms of server crashes (e.g., no input processing or output processing), poor response time, inability to handle multiple threads, or incorrect results? If the application is performing poorly because of limited server resources does that poor performance count against reliability metrics? How is reliability measured if the application is working in a degraded mode, but the server appears to be functioning? Without an explicit definition of a failure, organizations will have difficulty legally enforcing availability SLAs.

Maintainability is another important part of the overall availability of a system. Maintainability consists of the time it takes to identify that a failure has occurred, the time to isolate the cause of the failure, administrative and logistics lead times if parts or root access is required, the time to restore the system to operational capability, and the time to test the system to verify operational capability. In hardware maintainability can be improved through its design and documentation. The same is true for software.

An important part of the maintainability is the documentation. Accurate, timely documentation can mean the difference between meeting SLA and not. This documentation can include configuration data, documentation from the CRB, operating

procedures, recovery instructions, incident reports, monitoring information and trend analysis. It may be as simple as correct recall numbers of staff members.

Another very important part of maintainability is how well the backup tapes are documented and controlled. If the application is being backed-up correctly, and one week supply is kept on hand, the ability to restore a file or entire program is much quicker. The ability to quickly locate the correct tape and restore the necessary file depends upon proper documentation.

Maintainability is also dependent upon the skills and training of the staff. A well-trained staff will be able to isolate problems and repair them quickly. Additionally, good staff will be able to predict problems through trend analysis and good monitoring procedures before a failure occurs. A service provider may have the most reliable hardware and software available, but may not be able to meet availability SLA thresholds if their ability to correct problems is poor.

It is important that the program manager and the contractor define the concept of 'restored to operational condition.' The SLA should specify whether testing is required to validate restoration, or whether the contractor can make repairs and immediately return the system to its operational state. The SLA should also specify if someone from the program manager's staff needs to verify that the system was restored. The SLA needs to state the metrics that will be used to determine if the program is restored to operational condition. A method of determining the time the system went down and was considered restored also needs to be negotiated.

The SLA should also specify how planned maintenance will be addressed. Scheduled maintenance is predictable in that the time to perform the maintenance and restore the system to an operational state is known. Scheduled maintenance contributes to the downtime of the system, but some are reluctant to include scheduled maintenance in availability figures. Others feel that scheduled maintenance should be added into availability figures, as they are not able to utilize the system during the maintenance. Those that advocate not using scheduled maintenance are fearful that if they included scheduled maintenance time in availability figures, that the contractor will rush or skip

procedures to ensure that overall downtime was minimized. Either approach is acceptable so long as the SLA addresses the issue.

One of the problems with utilizing availability in SLAs is that the mean time to failure and the mean time to repair are estimates based collected data. In some cases enough historical data is available to calculate reliability and maintainability figures. In other cases, formal analysis such as a failure modes, effects and criticality analysis (FMECA) can be conducted by reliability engineers to estimate availability. In the case of new software, historical data may not exist. In other cases, estimates are suspect because of the small data sample size. In some cases it may be more appropriate to utilize confidence limits instead of a specific figure for determining availability in the SLA. (Wang) Another problem is that most estimates are based on ideal conditions, not on actual operational performance. Additionally, anytime new patches or versions of software are introduced, past historical performance may no longer be relevant. The same is true when software is operated in a new environment, or interfaces with new software.

The SLA needs to determine how availability measurements will be collected and applied. The program manager and the contractor will have to determine whether the measurements will be end-to-end, or whether specific components or pieces of the system will be measured. They will need to decide how many samples will constitute an accurate estimate of reliability and maintainability. The SLA will also have to define the time period over which the data is collected. A one-month period may be too small to collect enough data, and six months may be too long given the dynamic nature of most IT systems.

The SLA that pertains to host environment availability in appendix (A) takes a different approach. Because of the difficulty in determining a legally enforceable definition of a failure, and the difficulty in obtaining enough samples to evaluate whether availability thresholds were met, the author felt another approach was needed. If availability was defined in terms of an ‘opportunity to compute,’ then key server and infrastructure performance parameters can be identified, quantified, and measured. The SLAs can identify key performance thresholds that must be maintained for an application

to properly function. If the thresholds are violated, the application is considered impacted, and the service provider will be penalized accordingly. The SLA will also specify expected recovery times based on the severity of the impact. If the server, operating system, and infrastructure are operating within parameters, then the application should be able to perform all of its functionality. If the application is programmed properly, then by guaranteeing the appropriate resources and latency, the application should always be able to meet operational needs.

This approach alleviates many of the problems found with defining and measuring availability. This approach is more straightforward, and there are numerous tools that can monitor the key performance metrics. It is not however, without its own set of problems.

Utilizing an 'opportunity to compute' approach makes the assumption that server and network performance is a good indicator of whether an application will perform as expected. In the SLAs in appendix (A), the application was developed and is maintained by the government. In this case, it is a reasonable expectation that the application will perform given adequate resources and bandwidth. Although it is possible to have a poorly designed application fail even if it has all required resources and bandwidth.

Unfortunately, specifying the appropriate resource requirements to meet operational requirements can be difficult and will vary depending upon the type of server, the operating system, and the architecture being used. Network parameters are relatively straight forward, but server resources are more difficult to equate to application performance. Most system administrators have their own set of key indicators and thresholds to monitor, based on experience, skill levels, and the equipment they are utilizing. The metrics in appendix (A) are commonly utilized by the system administrators interviewed.

Approaching availability as an opportunity to compute also makes the SLA more adaptive to changes. Historical data on reliability and maintainability is not needed. In terms of availability, the Configuration Review Board (CRB) only has to evaluate any hardware or software changes or modifications in terms of the key performance indicators, capacity management, and documentation.

E. SUMMARY

The choice of quality factors depends upon the mission of the system, quality requirements from stakeholders, and the external environment. Part of the SLA development process is to identify mission critical business processes and determine those quality factors necessary to support those processes. Once the stakeholders have identified all of the quality factors, they must be prioritized and any conflicts must be resolved. The quality factors are also broken down into sub-factors, if possible, and assigned quality metrics that will measure the quality factors. The use of template SLAs can help identify various quality factors, but they must still be modified to meet the needs of each system.

Quality factors are not always easy to measure. The quality factor 'response time' is a good indicator of performance from the end-user's perspective, but it is difficult to obtain end-to-end measurements, especially if the host provider does not own the infrastructure. Response time can be measured at the server level using synthetic transactions, but this measurement has limited value to the end-user. Availability is also difficult to measure, as the contract must explicitly define downtime, statistical measurements are suspect because of the small sample size, and restore to operational condition must be defined. Measuring availability as an 'opportunity to compute' makes the measurements easier, and it accomplishes the same goal.

VIII. CONFIGURATION MANAGEMENT

This section will discuss configuration management in some detail. The detail is necessary to show the difficulty of managing software configuration, but it also demonstrates the areas where SLAs can be utilized. Quality factors can be established in the SLAs to ensure that proper procedures are followed, that the documentation is correct, that changes are being tracked, and that releases are managed properly.

Configuration management is an integral part of both development and maintenance of software. In its simplest form configuration management is how an organization manages change. However, a better definition is that configuration management is the discipline that ensures that the state of the software at any given time is known and reconstuctable. (Horch) Another more complex definition is that configuration management is the disciplined approach to managing the evolution of the software's development and maintenance practices, the resultant products and artifacts (data, tests, web content) and the processes involved in creating and changing them. (Dart) Configuration management can apply to software, hardware, and firmware, but this section will only discuss configuration management in the context of software.

The business environment is constantly changing as organizations attempt to gain competitive advantage. All projects will have changing requirements whether they are a result of external environmental pressures, new ideas, more efficient processes, changing technology, or corrections to problems encountered. Change is the one constant in any project. For example, from the time that the initial conceptual design was frozen to when the first production 767 rolled off the production line, 12,000 changes were made to the design. (Simpson) Good software engineering practices, as reflected in the CMM and IEEE standards, require a strong configuration management process to manage change. (Estublier, 2002) Organizations that cannot manage change will quickly have chaos.

Configuration management is incorporated throughout the software development and maintenance lifecycles. Configuration management captures information on every artifact (requirements, design, models, code), every action (edit, pass code to the QA department for testing, notify), and every person working on the system (developer,

tester, software engineer, program manager). (Dart) Some of the benefits of configuration management include better quality, dramatic productivity improvements, cost reductions, error/defect reductions, easier maintenance, and better technical support. (Leon) Other benefits include easier auditing, visibility into all work status, knowledge management, better forecasting and planning, and better adaptability to changes in business processes. (Dart) Unfortunately, despite the benefits, some developers feel that configuration management is just additional documentation and is not worth the extra work. Some developers are also willing to sacrifice configuration management in their rush to bring the software to market. SLAs can help to ensure that the contractor has an accurate and effective configuration management system.

Configuration management consists of four basic areas: configuration identification, configuration control, configuration accounting, and configuration audits. Another area of configuration management that is discussed in Appendix (A) deals with asset management, which is very important when dealing with recovery, maintenance support, trouble shooting, and disaster recovery. This section will also discuss the effects of configuration management on post-production maintenance activities.

A. CONFIGURATION IDENTIFICATION

IEEE Standard 828-1998 defines configuration identification as a process of selecting the configuration items for a system, and recording their functional and physical characteristics in technical documentation. Configuration identification also includes the process of uniquely identifying the version or instance of every configuration item (documentation, models, files, tests, specifications) that makes up or supports a software product. These items can also include the tools that were used to create or modify the software such as the HTML editor, Java interpreter, modeling tools, and code generator. (Dart) These configuration items can refer to versions of the entire system, modules, or they can refer to the smallest units of code that can be compiled. Each item needs to be identified and described so the organization has knowledge of its existence, its status, its interrelationships, its dependencies, and the effect that changing it will have on other items and the system.

One of the first steps in managing software configuration is determining what constitutes a configuration item. If every grouping of code that is capable of being compiled is included in the configuration management process the administrative efforts to document the code, report and analyze changes, and track status can overwhelm the developers and management. One the other hand if the level of abstraction is at the module level, it may not provide management with enough documentation of the subroutines contained within the module. At the module level, any changes within the module will require testing of the entire module instead of the individual subroutine that was changed. Selecting the level of decomposition at which to apply configuration management is important and can depend upon many factors such as size of the project, importance of tracking changes at the lowest levels (safety or timing issues), whether the item is standalone, new technologies, interfaces, requirements volatility, complexity, and risk aversion.

Another important decision is what information needs to be collected on each configuration item. Ideally all characteristics of the configuration item is collected to include its content, the documents that describe its function, the requirement that it is satisfying, data needed for operation of the software, the different versions as the software is changed, interface information, dependencies, and any other information that makes the software what it is. (Leon) However, the type of project will dictate the data that needs to be collected on each configuration item.

As each artifact or documentation is developed, reviewed, and approved, it must be included in the configuration management repository where it is assigned a unique identifier. When the configuration item is first entered into the repository, it is considered baselined. A baseline is a configuration item that is frozen in time to represent a specific state of a product. (Dart) Items that are in the process of development can be changed quickly and easily, but once they are baselined in the repository it must go through a formal process before it is modified. Once modified, it is assigned a unique identifier, so it can be distinguished from its earlier version.

The task of assigning a unique identifier has been made easier by a number of good automated configuration management tools. These tools ensure that a standardized

methodology is applied to assigning the identifier. Simple identification codes will include information on the parent or next higher component, when the item was created, and the version number of the item. More complex identifications include the project number, project type, item type (document, program, data, test), relationships, dependencies, release, version, and edition. (Horch).

The final step is to store the configuration item, documentation, and execution software (operating system, compilers, tools) in a secure repository where the item can be retrieved and reproduced when required. This is especially important when software needs to be rolled back to a previous version, or when software needs to be reinstalled to correct problems.

SLAs can be written to specify quality factors that deal specifically with the accuracy of the configuration identification and the information collected on each configuration item. SLAs can also be written to verify the accuracy of the repository to ensure configuration items can be recovered if needed.

B. CONFIGURATION CONTROL

Configuration control consists of those processes necessary to ensure that every change to a configuration item is reviewed, authorized, tracked, and documented. Once an item has been baselined, more formal procedures need to be instituted to ensure that only approved changes are made to an item. Changes need to be reviewed to determine their relevance, their impact on other configuration items, and their impact on cost, schedule, and performance.

A software change order may be needed for a number of reasons including the need to rework a component with poor quality, the need to rework a component to achieve better quality, or because of a user directed change in requirements. The first two types of change need to be closely tracked as they are indicators of the quality of the product, and they provide a solid basis for estimating maintainability. (Royce)

Configuration control also provides a documented evolution of how and why the file or module evolved to its present form, and the changes that were made along the way. The history of changes on a configuration item helps personnel understand why changes

were made, it helps with trouble shooting, and it helps maintenance personnel determine why specific changes were made.

The goal of configuration control is to prevent 'guerrilla programming,' where developers are making changes to software without considering the effects that those changes will have on overall functionality, quality, or other configuration items. Configuration control ensures that changes are documented, analyzed, incorporated into the schedule, tracked, tested, and incorporated into user documentation. Configuration control also ensures that only known and approved changes are being worked on which helps focus the work effort on those areas that provide the most utility. Configuration control also helps to avoid situations where developers are working on 'nice to have' or unspecified functionality that they think the user might need.

Configuration control can be broken down into four slightly overlapping areas. The change review board reviews proposed changes to evaluate their need and their impact. Change management is concerned with tracking the status of the change. Notification is the process of keeping programmers informed about changes that impact their area of responsibility, and release management is concerned with releasing and tracking updates and patches to a baseline configuration. Quality factors can be specified for each area, and they can be incorporated into SLAs so their respective quality metrics can be monitored.

1. Change Review Board

Configuration control starts with a change request form. In most cases this form is now automated and is a part of the configuration management software package. The change request form identifies the configuration item to be changed, it describes why the change is necessary, it describes the type of change, it describes the priority of the change it describes what changes will take place, and it provides an impact analysis. The impact analysis evaluates whether any other configuration items will be affected by the change and what actions will have to be taken in those configuration items. The impact analysis can also look at how long it will take to effect the changes, their costs, and the benefits. The change request form is often initiated from a software trouble report. Once the

configuration item is baselined, a change request form should be utilized, as it has to be approved by the change review board.

Once a change request has been submitted, it is passed to the change review board for approval. The change review board (CRB) is tasked with evaluating the change requests and determining whether they will be approved, delayed, or denied. The change review board also monitors the progress of every approved change. The change review board also determines which reported defects to correct, and when they should be corrected (what release).

The change review board should consist of the configuration manager, the program manager and members of that team (especially contracting personnel), developers, the test community and quality assurance, marketing, and essential stakeholders. The head of the change review board should be the configuration manager as that person best understands the need for configuration control, and that individual is typically impartial, and does not have an agenda other than enforcing configuration mandates. (Harris) The CRB is designed to make informed business decisions regarding all proposed changes, which will provide the greatest business and customer value while controlling the system's lifecycle costs. (Wiegers)

Depending upon how the configuration management process is implemented, change requests may include impact statements, or they may be ordered after the CRB makes an initial determination as to whether the change is warranted. Before approving a change request the CRB needs to analyze the change with respect to the effect the change will have on functionality, the impact on other configuration items, and how it will impact cost and schedule.

The CRB must first determine whether the change is necessary. The change request form should contain the information necessary to make a determination. If not, the form will be returned for further information. The CRB needs to evaluate the criticality of the change and determine whether it should be implemented in the current release (which will probably impact schedule), whether it is delayed (the change is incorporated into another release), or whether it should be rejected (the change was a result of an unauthorized request, the impact to the system was negligible). Changes that

are submitted to fix errors or improve quality need to be weighed against the benefits that those changes provide. If the package meets requirements, but can be made better, the CRB must decide whether the change is warranted given other considerations such as time, money, goodwill, and lifecycle costs. New requests must also be evaluated in terms of when they will be incorporated into the release. Many projects have failed as a result of being unable to maintain a release baseline. At some point changes need to be deferred to future releases or the baseline release will never be fielded.

It is important that the CRB determine what types of changes need to be reviewed, and which can be automatically authorized (automated) or referred to a lower level manager. Minor changes still need to be logged into the configuration management system, but they do not need the attention of the CRB. If the change approval process is too stringent, programmers will discover ways to circumvent the procedures.

The CRB also needs to review the changes to ensure that they do not adversely impact any requirements. All proposed changes should be linked to the requirements that the configuration item satisfies. The CRB needs to ensure the test community incorporates the revised configuration item into the test plan to ensure performance and functional requirements are met. The CRB must also take a holistic look at the impact the change will have on SLA mandated non-functional quality requirements. New requirements must be reviewed to ensure they do not conflict with functional or non-functional requirements. Any conflicts will have to be resolved by the program manager, stakeholders, and the contractor.

A good configuration management system will specify the other configuration items that interact with the file or module that is being changed. The impact analysis will determine the amount of work necessary to modify those configuration items that are affected by the change. A small change in one file or folder may cause a great deal of change in other areas. The changes must also be reviewed to determine their impacts on the software architecture and supporting models that will need to be updated.

The CRB must also evaluate the changes with respect to costs and schedule. New requirements may require revisions to both costs and schedule. A contracting person from the program management office and the contractor should be part of the CRB to

ensure that contract modifications are drafted and approved before any changes are approved that will affect price. Depending upon the requirements, SLAs may need to be revised.

If SLAs are utilized in a contract, a CRB must be established to ensure that any proposed changes do not impact the quality factors specified in the SLAs. Since the SLAs are contractually binding any unauthorized change that impacts that contractor's ability to satisfy a quality threshold can, in a worst case scenario, result in legal proceedings. In most cases, the change will have to be reengineered so it will not impact the quality threshold. If the change still impacts the SLA, then contractor will not be held accountable for meeting the SLA requirements, and new SLAs will have to developed and negotiated. The lack of a CRB or a similar process will quickly undermine all of the efforts to establish the SLAs and will make them worthless.

2. Change Management

A good configuration management system is capable of tracking every phase that a change request goes through (the change request form, the impact analysis, results and comments from the CRB, task assignment, the new or modified code, test, acceptance, and assignment of a new configuration identification). (Dart) The CRB is responsible for tracking and maintaining status on the configuration items that have been approved. Although most of those tasks are automated, the information still needs to be entered into the system. Each time a change goes through a phase, that information needs to be captured in the configuration repository.

Another function of change management is coordinating the work on a configuration item. The configuration manager or software librarian generally controls this function. One of the main functions of configuration control is to coordinate the access to and modification of configuration items when multiple people could be working with the same configuration items. (Sarma) One approach to avoiding having multiple people modifying the same file or folder is when authorized changes are approved, the developer copies the file or module to be modified, and sets a lock on that file (check-out) so another programmer does not make concurrent changes to the same file. Only the authorized programmer is allowed to create a new version of the file (check-in). (Mei,

Estublier, 2000) Part of the control process is defining who has authority to perform a specific change, when that change can be performed, and what changes can that individual make. Controlling concurrent programming or distributed programming can be difficult, but lack of control can be disastrous.

Change management also includes risk analysis. The CRB and the program manager need to assess the risks associated with introducing new requirements at either the system level or the software level. At NASA they use several factors to assess that risk, including the size of the change, the location of the change, its criticality, the number of modifications, and resources needed to make the change. (Schneidewind 2001) The program manager needs to carefully monitor the amount of new requirements that are generated during development. It is very difficult to limit changes to a baseline version (political factors, changing business environment, new ideas), but there has to be a cutoff point where additional changes are moved to later versions. High requirements or change volatility throughout the initial stages of development indicates that the stakeholders do not really know what they want, or the development effort was more difficult than anticipated. In either case the risk to the success of the project increases with change volatility.

3. Notification

The Lantau Airport Railway project was a complex system of systems project to build a railway from the airport to the urban areas in Hong Kong. It was a seven year project that consisted of over 40 contracts. The command and control system and the billing system accounted for the majority of the software. One of the major problems that they encountered was a failure to communicate changes among all of the contractors. As the lifecycle of the project matured they discovered that the contractors would make small changes to the interface specifications. These changes were not always communicated to other contractors that may have to interface with that system. This was due in some cases to time differences in development schedules, and the lack of a central repository for all contractors. (Wong)

To coordinate access to a common set of configuration items by multiple programmers working on the same project, most configuration control systems utilize

workspaces (part of a file system where the file of interest is located) where the developer can work isolated from the outside world and other developers. The workspaces support concurrent engineering in two ways. The first is controlling who has access to the workspace, and the second is resynchronizing (merging concurrent changes to the same file) where algorithms can identify changes to the file and blend them into one file. (Estublier,2000) Control can be accomplished by locking files (which forces serial development) or concurrent changes and resynchronizing can be utilized. Unfortunately, the workspace does not allow developers to know what changes are being made in parallel to their efforts as they cannot see into other workspaces. Configuration management systems are still struggling with concurrent development issues and notification, although there is some good research in this area. (Sarma, Estublier, 2000)

Despite the notification problems at the working level, configuration management systems are able to identify at a higher level, those configuration items that will be changed, and what the changes will consist of. The difficulty is determining how to convey that information to the developers and the stakeholders. Notifying all of the people that need to know about an approved change is a process that needs to be planned, controlled, and monitored. It is also important to note that the software CRB has representation on the system CRB, so as system changes are made, the appropriate people are notified, and the system changes are incorporated into change requests at the software level.

The configuration management system also needs a method to notify users of the status of their change request. Users need to know whether their request has been review, whether it was accepted, who was assigned the work, and when the change will be incorporated (what release). Some management systems have an e-mail notification that lets them know when their request was reviewed.

4. Release Management

Large organizations also have a representative release committee, which controls the content and timing of releases. The release committee is responsible for coordinating releases with the stakeholders. All projects have stakeholders with different agendas, priorities, and beliefs concerning how the project should be run. The release committee

works with the stakeholders to achieve some form of consensus concerning the functionality that will be incorporated into the baseline and future releases. The release committee also tries to ensure that all stakeholders have consistent information regarding what functionality will be included in the various releases. (Dikel)

Another part of configuration control is monitoring which release stakeholders are using. While this appears to be straightforward, it is not. It is not uncommon for multiple versions of the same software to be deployed by various stakeholders due to beta versions, unique functionality integrated into a specific version for a particular stakeholder, failure of the system administrators to load the new version, lack of resources to run the new version, or failure to receive/download the new version. It is also important to know what version of environmental software (the operating system or database management system (DBMS)) stakeholders are using. Changing environmental software can be extremely time consuming as all applications and tools residing on the current operating system will have to undergo regression testing before migrating to the new operating system. The Navy and EDS discovered how difficult that was when they migrated applications into the NMCI system.

Coordinating version releases can be very difficult, especially when they interface with legacy applications. The move to Oracle 9I may have a huge effect on some of the older systems. In addition, the applications will have to be thoroughly tested to ensure that they are compatible with the new DBMS. Some applications will have to be reengineered. This will require time, money and manpower, all of which are in scarce supply. This gets even more difficult with distributed systems that reach back into old databases that may not be under the control of the program management team.

C. CONFIGURATION ACCOUNTING

Configuration accounting is process of tracking and reporting the status of all versions of the software (from the configuration items to the entire software system), models, architectures, documentation, and change requests. Configuration accounting starts with determining the baseline of the software system. This is normally done during the major reviews that mark the end of a lifecycle phase such as the software

requirements review (SRR) or the CDR (critical design review). The baseline can also be established once a package of configuration items has been tested and approved.

Configuration accounting ensures information regarding the baseline (date, who approved it, how it was established) and any subsequent changes is captured. Configuration accounting maintains records regarding the change request, actions of the CRB, status of the change request, status of the change, the expected completion date, and the assigned release number. Another purpose of configuration accounting is to ensure that the name, release, version, and edition of each configuration item, and each of its subordinate items are recorded, monitored, and when necessary updated. When changes are made, the configuration identification of all affected configuration items must be updated. (Horch)

Configuration accounting should support queries such as how many change requests are pending CRB review, how many changes have been rejected, the number of change requests in a particular module, as well as a breakdown of the type of requests. The configuration management system should also be a useful management tool in that it should be able to track all change requests that are in progress (being developed, awaiting testing, in testing, awaiting approval, completed and assigned new configuration identification) completion dates for those changes, how many changes are pending for a future release, the priority of the change, and which changes are not meeting schedule.

D. CONFIGURATION AUDIT

Configuration audit is the area that SLAs have the most utility. Configuration auditing is the process of keeping an audit trail of all actions, events, notifications, and testing that happened to a configuration item. Configuration audit also constantly monitors the configuration management system to ensure that at any time configuration items are accurately identified and that the configuration management process is working correctly. (Dart)

Establishing a good configuration management system can be very time consuming and the tools are expensive. Unfortunately, the system is only as good as the people running it and the information that is being fed into the system. If the information

in the configuration repository is not accurate or lacks the necessary information, then the systems usefulness as a quality control tool can be questioned. The system must be audited to identify areas that may need more attention or training. Additionally, auditing can also determine if the right changes were made to the configuration item by comparing the change request form to the documentation that was provided as part of the item's modification. A quality software product is dependent upon an accurate configuration management system and process.

SLAs help the program manager audit the configuration management system through the use of quality metrics and the monitoring process implemented by the SLAs. SLAs can specify that configuration identification accuracy on weekly spot checks must be 98 percent and the accuracy of the accompanying documentation must be 95 percent. Spot checks can also determine the effectiveness of the CRB in controlling changes. SLAs can specify that of the changes that need to be reviewed by the CRB, 99 percent of the changes must have been reviewed by the CRB. Similar quality thresholds can be applied to documentation requirements, notification procedures, configuration accounting accuracy, change management procedures, and audit trails.

E. ASSET MANAGEMENT

Program managers also need to maintain tight configuration control in the host environment once the application or system has been fielded. Appendix (A) includes threshold values on the accuracy of the configuration management system in the host environment. Accurate configuration data is essential for troubleshooting, disaster recovery, and it is an important element in capacity management.

Accurate information regarding the hardware and environmental software that is hosting the software system will help evaluate the effect that changes to the software or environmental software will have on the system. If developers are writing the program using the fastest available PCs, their users may experience performance problems because they are using PCs that are two or three generations old. If distributed sites are using different firewalls and have different restrictions regarding port utilizations,

problems may occur. Troubleshooting and planning will be easier if there is enough information concerning all hardware and software assets in the host environment.

Asset management is critical during disaster recovery, especially is a cold site is used. If new equipment needs to be procured and installed, knowing the type of equipment being used, the infrastructure and network configuration, environmental software, and system software is critical. Small errors in the versions of software being utilized can take hours of troubleshooting to resolve. Good configuration control will also help to ensure the proper files are restored in case of problems. Installing the wrong file can have disastrous effects.

Capacity management ensures that the IT infrastructure is capable of supporting the computing demands of the systems being supported. In the post-production phase the change management process should also identify the performance requirements associated with each change. Any changes (modification or new requirements) to the software may also affect the infrastructure in terms of throughput, performance, port utilization, security, CPU utilization, memory usage, response time, and availability. For example a new requirement to encrypt any e-mail notifications that the system generates may impact the performance of user's PCs, internal network performance, or it may require modifications to the firewall. The configuration repository should be updated to include the technical specification for each change item (e.g., disk space, speed of processor, expected workload, demands on IT services). New requirements may necessitate negotiating new SLAs. (ITIL)

The CRB does not go away after a product is fielded. Maintenance of the software needs the same configuration controls as development, or the fielded system will quickly develop problems. Program managers need to understand the implications that maintenance actions are going to have on their systems. They also need to assess how changes in the system requirements or architecture will affect their entire system.

F. SUMMARY

Configuration management gets little attention if it is done correctly, but if it is done poorly, the entire development and subsequent or maintenance process suffers, cost

and schedule predictions will be underestimated, and the defect rates will increase as programmers make changes that affect other artifacts. Program managers can utilize SLAs to monitor the contractor's configuration management procedures and accuracy. SLAs reduce the risks associated with poor configuration, and they promote quality throughout the configuration process.

THIS PAGE INTENTIONALLY LEFT BLANK

IX. PROGRAM MANAGEMENT

Organizations are increasingly relying upon information technology to enable their critical business processes. Despite the increasing complexity of today's systems, organizations are demanding extremely high levels of quality in the IT systems that they are acquiring or producing. In many industries the efficiency and effectiveness of an organization's IT systems is what gives them a competitive advantage in the market place. Poorly performing IT systems can result in lost market share, lost customers, and lost opportunities. As a result, upper management is placing great pressure on program managers to deliver quality products.

It requires a great deal of management to produce quality software. Program managers have to ensure that quality considerations are addressed early in the lifecycle and they must provide the proper amount of oversight to ensure those quality factors are incorporated into the final product. One of the major difference between a software project manager and other areas of management is that the software project manager must not only understand the intricacies of management (requirements, planning, budgeting, contracting, oversight, tracking), but they must also understand all aspects of the software-development process, as well as understanding the application domain for which the software is being developed. Unfortunately, there are not many program managers that have the software experience necessary to effectively manage a large software intensive project.

Service level agreements can assist program managers in many of the tasks necessary to ensure quality is delivered in the final product. SLAs are particularly useful in the areas of risk management, financial management, contract management, quality management, and customer satisfaction.

A. RISK MANAGEMENT

A risk in the context of program management is a potential event that can adversely affect the project. Risk management is the proactive process of identifying and mitigating potential risks throughout the lifecycle of a system. When developing

software there are many types of risk that have the potential to affect the project such as product risk (the system may not meet expectations), project risk (cost and schedule), financial risk (another investment may provide more benefit), business risk (the system will not generate expected competitive advantage), and technical risk (design, interfaces, compatibility). The program manager is responsible for developing a risk management plan to deal with each type of risk. SLAs can help to identify risks in the requirements engineering phase, they can mitigate risks through the use of standards and performance monitoring, they provide valuable input to the test plan, and they help manage risks in the post-production phase.

Another categorization of risk proposes that there are three types of risk, known risk (can be discovered after careful evaluation), predictable risk (based on past performance and lessons learned), and unpredictable risk, which are very difficult to identify in advance. (Pressman) Senior management and stockholders of the organization expect that the program managers will take all necessary steps to address the first two risks. In the government, program managers have to submit their risk management plan to the director of the Office of Management and Budget (OMB), as OMB has been tasked with analyzing, tracking and evaluating risks and results of all major capital investments in information systems. (Clinton) The government and industry realizes that failure to address risks can have serious ramifications. The result of project failure can result in fiscal loss, a loss of reputation, loss of market share, damage to the brand name, and a loss of competitive advantage. (Frost)

Although the program manager is generally tasked with risk management, it is a team effort that involves the input of all stakeholders. Risk is a subjective notion, and it is important that risk, from the perspective of all stakeholders is examined. It is also very important that the program management team understand the level of risk that upper management is willing to take regarding the program. Factors such as the maturity of the company, its financial stability, its portfolio of other programs, and the expected return on investment all influence the level of risk management is willing to accept.

The program manager needs to take a holistic look at risk management. Risks need to be identified to the greatest extent possible at each stage of development and at

multiple levels of abstraction from the system level to component design. It is also important to realize that risk management involves uncertainty and the intent of risk management is to take actions that reduce risk to levels that management is willing to accept. It is not possible to eliminate all risk.

The risk management process generally consists of five steps. The first is to properly scope the project and determine the risks associated with the project. The next step is to analyze the risks to determine their impact, identify factors that will affect those risk areas, and evaluate the likelihood of occurrence. The third step is to prioritize the risks. The next step is to determine a course of action that will mitigate the risk if possible. The final step is to monitor the effectiveness of the risk mitigation plans. (Peltier, P. Smith) Each phase of the development cycle will contain risks unique to those phases, but the impact of those risks has the potential to affect the entire project.

In the requirements phase, risks are evaluated in terms of the extent to which stakeholders can define what they want the system to do, project size, technical feasibility, interoperability concerns, project cost and schedule, and the effects the system will have on the business processes it supports. In the development phase, the architecture, design, code, requirements churn, and processes are evaluated to determine whether the system will be delivered with the required functionality and quality within budget and schedule. Once the system is deployed risks are analyzed in terms of customer satisfaction, resource availability, maintenance actions, disaster recovery, and configuration management.

1. Risk Management in Requirements Phase

The first step in the development of a risk management plan is to scope the project and identify the risk drivers. Most organizations utilize a risk identification checklist that is developed from industry standards, benchmarking other organizations, or they are internally developed to incorporate a specific organizational culture. The checklists consist of primarily predictable risks, but they also include some known risks. The risks are then ranked based upon the probability of occurrence. Then next step is to analyze the impact that the risk, if it occurs, will have on the project. The risks can then be assessed to determine impacts on cost, schedule and performance. A risk management

plan can then be developed to mitigate the occurrence of risk, monitor risk areas, and reduce the impact if the risk occurs. Although risk management occurs throughout the lifecycle of a system, much of the plan is developed during the requirements engineering phase. The SLA development process contributes to the development of the risk management plan by improving communication between stakeholders, challenging assumptions, prioritizing risks, identifying risks, and proposing steps to mitigate risks.

Before the project is even started management must determine whether they should invest the time, resources, and capital in the system. Management must evaluate their customers, employees, competitors, available resources, and the environment to determine where they should invest their capital to obtain the greatest return or position themselves in the market to obtain a competitive advantage given a dynamic business environment. Some of the risks in the concept phase of the project are whether the system will return the benefits expected, whether other projects could return more benefits, whether the project can be completed in time to leverage its capabilities for financial gain, whether new technology will quickly make the investment obsolete, whether new partners will be able to interface with the system, and whether the end users will embrace the system.

If the concept is approved, the program manager must first determine the proper scope and of the system. When defining the scope of the project, the program manager must determine what functions the system will and will not perform.

Some systems are inherently more risky than others. Systems that utilize existing technology to support low value business processes are not as risky as systems that utilize complex or emerging technology to support a critical business process.

Before specific requirements are gathered, the program manager should already be considering general risks associated with interoperability considerations, the operating environment that the system will be deployed in, whether emerging technology will be utilized, the skills of the management team, the experience of the contractor or in-house developers, schedule and cost constraints, the size and complexity of the projected system, and the affects of a dynamic market place.

During the requirements engineering phase, the scope of the system will be refined, and a better understanding of the requirements will lead to more risk identification. In addition to risks associated with the system, there are also risks associated with the requirements engineering process itself. Some of the common program risks associated with requirements is whether customers were involved in the requirements engineering process, whether stakeholders have realistic expectations, whether requirements are stable, and whether the requirements are complete. (Pressman) The SLA development process addresses many of the requirement risks.

Risk management tries to reduce the amount of uncertainty as much as possible. The SLA development process in beneficial in bringing stakeholders and the contractor together to discuss project scope, assumptions, functional requirements, as well as non-functional quality requirements. Risks can be reduced by gathered as much information as possible concerning stakeholder and management's expectations in terms of system functionality, performance, costs, schedule, and budget. The process of developing SLAs fosters communication among stakeholders and will serve to identify many assumptions and make explicit many implicit requirements. The development team can provide the program management team with a great deal of information to reduce some of the uncertainty.

The development team consisting of individuals with different backgrounds and perspectives can also help the program management team in identifying risk areas that the program management team did not consider. Many risk identification checklists do not include non-functional requirements, despite the fact that there are many risks associated with those requirements. Template SLAs can also help to identify risks.

The program manager must also evaluate the assumptions associated with the system. Some of the assumptions include the amount of support management is willing to give the project in terms of talented workers, resources, facilities, and power. Other assumptions include the degree to which requirements are known, whether all stakeholders have been identified, whether new technology will be mature by the implementation date, whether COTS packages should be incorporated into the system (Schneidewind 1998) and whether internal and external business trends will continue.

Assumptions should be evaluated in terms of the degree of uncertainty, possible impacts, whether they are valid, and how they will be addressed.

The SLA development process is also helpful in defining and prioritizing those business critical processes that must be supported in the new system. Identifying critical processes allows the program manager to concentrate risk management efforts in those areas. In a large project it is very difficult to manage the all of the risks that have been identified. Efforts need to be focused on those areas that have the largest potential to cause damage, or that have the highest probability of occurring. Resources are too scarce to waste effort on low risk areas.

Identifying critical processes also helps in assessing the security requirements and risks to the information used, processed, and sent from the system. The efforts spent protecting the information in the various pieces of the system has to be weighed against the business criticality of that information and the processes they support. Stringent security requirements provide more protection for the information, but they also make the system less flexible. SLAs that deal with security focus on those critical information areas.

SLAs can be utilized to mitigate and monitor product and process risk. Depending upon the risk identified, SLAs can be developed to establish quality thresholds for that area. For example if one of the risks identified is in the schedule planned for the project, then measurements can focus on total project effort, aggregated schedule slippage, project staffing, requirements churn, critical path analysis, size (i.e., COCOMO II), and complexity. The monitoring process and reports generated as a result of SLAs focus management and the contractor's attention on the areas covered by the SLAs.

SLAs can also be used to encourage the contractor to devote additional attention to risky areas through the use of incentives or penalties. If schedule risk is a high priority, then incentives can be offered if the actual schedule is better than the estimated schedule. In determining what to measure it is helpful to determine the behavior you want from the contractor, and determine what measurements will most likely encourage that behavior. (Kendrick) The SLAs mandate monitoring of the quality factors associated with process

and project quality. If quality thresholds are not met, program managers and the contractor are informed of the violation, and the program manager is at least aware of the increased risks associated with that particular quality factor. That knowledge may lead to closer monitoring or corrective action to reduce the risk and improve the quality.

Project managers should review risks identified in prior projects for lessons learned. Evaluating risks identified in prior projects, remediation actions taken, and their effectiveness can offer valuable insights. Risk management is easier when common processes and procedures (i.e. standards) are utilized. Historical data can be gathered and statistical analysis can be applied to new projects. Applying historical data on projects that differ in processes and methodologies is more difficult and less accurate.

SLAs can be utilized to ensure management and contractors understand the standards to be used in the project. As discussed earlier standards SLAs will also ensure that the project is monitored to ensure that the specified standards are being implemented correctly. Deviations from prescribed standards are an indication that the software-development process is veering away from the production of quality software. (Horch) Test plans can also incorporate audits of processes to measure contractor compliance.

2. Performance Monitoring

Performance management reduces overall program risks by ensuring that mission critical services, processes and procedures are being followed. A good performance management plan will help the contractor identify potential problems throughout the system's lifecycle before they result in loss of business functionality. SLAs support performance management through performance data collection, real-time monitoring, problem detection and diagnosis, and trend analysis. (Simitchi)

To reduce and manage risk, program managers need to measure or monitor contractor and system performance throughout the project's lifecycle to ensure requirements, standards, and quality factors are being met. Monitoring performance, whether through progress reports, milestone reviews, real time software monitoring, audits, or formal inspections, serves to inform the program manager of potential problems

(risks), it allows the program manager time to take corrective action, it influences contractor performance, it provides information for future projects, and it helps to achieve a higher quality product.

The program manager must develop a plan or methodology to determine whether the contractor is performing in an effective (requirements are being met) and efficient (economical utilization of resources and time) manner. Program managers cannot simply place requirements in a contract, award the contract, and test the final product to determine compliancy. The risks and potential for failure are too great using that approach. The plan must also cover system performance to ensure that it is operating within specifications.

Performance management is a process whereby the contractor is given concise quantitative requirements, feedback mechanisms are put in place to evaluate compliance with the requirements, consequences for noncompliance are discussed, contractor behavior and subsequent performance is monitored, and actions are taken by both parties if problems persist. (Richman, De Waal) Part of the SLA development process is identifying those functions, quality factors, standards, and processes that are critical to ensuring the delivery of a high quality product. The SLAs not only specify the quality factors and thresholds that need to be adhered to, but they also specify the means and timeframes to measure compliance with those quality thresholds, they establish resolution procedures, and they contain penalties for noncompliance. The program manager can utilize the information contained in the SLA as part of the overall performance monitoring plan.

Analyzing the data collected from monitoring can identify trends that can also reduce risks. Most SLAs require periodic as well as real time reports that provide performance information on the system. By regular monitoring and comparison against SLA thresholds, exception conditions can be defined, and near misses of SLAs can be reported upon. For example, analysis of monitoring data may identify issues such as contention (data, file, memory, processor), inappropriate locking policies, inefficiencies in the application design, unexpected increased in transaction rates, and inefficient use of memory. (ITIL) The data can also be used to modify the SLAs if necessary, predict

future resource usage, or evaluate the SLAs in terms of their effectiveness in reducing risk, improving software quality, and driving contractor behavior.

Template SLAs specify quality requirements in many of the common critical success areas (e.g., if the results obtained in those areas are satisfactory, the project will be successful). Although template SLAs have to be tailored to each project, they are useful in that they may highlight areas that other development teams felt were important to the success of their system. Contractors are more likely to devote effort to areas that they know will be inspected. As such, SLAs are useful in focusing the contractor on processes, procedures, and designs that will reduce risk and improve quality.

Performance monitoring should also apply to the host environment. In addition to monitoring system performance (throughput, resource utilization, response time) the program manager should monitor infrastructure performance (jitter, latency), security, problem response, end-to-end quality metrics, and availability. Risks are reduced by monitoring the entire spectrum of the system because problems can be quickly identified and resolved, trend analysis can identify potential problems, and a holistic view of the system may identify end-to-end risks that were not seen by monitoring system performance only.

3. Test Plan

A good test plan helps to reduce product risk. Managing risk attempts to reduce the amount of uncertainty as much as possible. A well-developed and executed test plan can assist the program manager in reducing some uncertainty. The purpose of testing is to validate that requirements have been met and to discover problems or defects. Reviews, inspections, and testing can create a great deal of information on the performance of the system and the contractor. Testing provides confidence in the system, it provides an additional perspective on risk, and it reduces overall product risk.

The additional personnel that conduct the testing also help to reduce risk by bringing additional skill sets and perspectives to the analysis of the module, architecture, system, or processes. Additional input from the test community can be helpful in identifying problems and developing solutions or better processes.

Testing gives the program manager a certain amount of confidence in the product, and in the contractor's ability to deliver a quality product. Much like performance management, testing allows the program manager to measure the level of success in achieving specific critical success areas. If a contractor is not performing well in unit or module testing, then the overall risks to the project being completed on budget and on time increase. Testing in the early phases of a project allows the program manager to take action to resolve the risks.

Testing also reduces risks by discovering defects before the project transitions to operational status. Risks to schedule and budget increase the longer a defect remains undetected in the system, as it is much easier to correct deficiencies in the beginning phases of a project. (Horch) A rigorous test plan reduces risk that is passed on to the customer in terms of functional problems, software safety and security, and user dissatisfaction.

The previous chapter discussed how SLAs can help program managers in the development of the test plan by helping to identify critical quality factors, increasing communication with the test community, quantifying quality thresholds, and defining how the requirements would be verified.

4. Post-Production Risk

The program manager is also responsible for managing the risks associated with post-production support. In post-production support the program manager is not only concerned with the performance of the system (meets functional and non-functional requirements), but they must also be concerned with the risks associated with the host environment (facilities, servers and infrastructure), the communications channels, and follow on maintenance actions. SLAs can be written to address many of the post-production risks including physical security, problem resolution, disaster recovery, and security.

One of the risks that the program manager must address is the physical security of the host environment. Physical security is not only concerned with employee access, but it also deals with issues such as whether the data center has fire detection and suppression

systems, the condition of the electrical grid, whether water pipes run through the data center, the condition of the heating and air condition system, and the amount of dirt or dust in the air.

The system can be designed with great application security, but if unauthorized employees or maintenance personnel have access the data center or tape storage area, then those application security measures can be easily bypassed. The data center should be restricted to only those personnel that must have access to perform their daily work, access to secure areas must be protected by an electronic access control system, and security must be monitored 24 x 7. The security system must also have a log of when employees accessed those secure areas for auditing purposes. Appendix (A) lists a number of physical security requirements in the facilities requirements section. When security procedures and processes have been agreed to, SLAs can be used to ensure those processes and procedures are adhered to.

The availability of a system depends in part on the speed at which the system can be restored once a crash has occurred. If files or programs need to be restored from backup tapes, then those tapes need to be quickly accessed, and they must be accurate. The risk that the system will not meet availability goals increases if the host provider does not have good backup and tape management procedures in place to ensure that all system software and related storage configuration can be recovered if an operational or hardware failure occurs. Appendix (A) contains a number of backup and recovery requirements. SLAs can be utilized to ensure agreed upon procedures and documentation requirements are being implemented correctly.

The program manager must also evaluate risks in terms of a natural disaster or terrorist attack. The host provider must have a disaster recovery plan to cover the possibility that a hurricane, tornado, flood, or blizzard damages its ability to operate for an extended period of time. Disaster recovery, or business continuity involves the planning and implementation of procedures to ensure critical business operations resume following a disaster and that they return to normal operations as soon as possible. Part of the process is determining which applications are critical and which are not, then deciding upon the time frames for recovery and site recovery necessary to meet the

recovery needs. In most cases organizations are too dependent upon their IT systems for their core business functions to loose that functionality for more than a couple of days. Some organizations cannot afford to loose their systems for more than a couple of hours.

Good disaster recovery plans utilize backup sites that are not in the same geographical proximity to the data center. Appendix (A) describes three types of backup sites that are commonly used, shell sites, warm sites and hot sites. A shell site just provides the necessary facilities for computing, it does not provide any equipment. A warm site provides facilities and equipment, but all system software would have to be installed on the equipment. A hot site provides facilities, equipment, and system software, which receive backup data from the host site at least daily or depending upon the criticality of the system in a near real time. The hot site should mirror the system in the host environment to the greatest extent possible.

Good recovery plans should have a disaster recovery team listed with cell phone numbers, a blueprint of where equipment and infrastructure are located, a list of vendors to call to replace equipment and software, a complete inventory of the hardware (model numbers, purchase date, associated software with version numbers), a complete inventory of the software (version numbers, licenses, license keys, date purchased), maintenance contracts, all relevant phone numbers (especially the recovery site), installation and operating procedures for the hardware and software, and personnel requirements to recover the existing site and run the remote site. (Philcox) The recovery plan must be exercised periodically to ensure the host service provider can provide recovery in the time frames stipulated in the contract or the SLAs.

SLAs help to reduce risks by identifying risk areas and proactively monitoring development processes and procedures and system performance to identify problems before they become serious. The SLAs also encourage the adoption of standards, which reduce risk, increase effectiveness, and standardize operations throughout the organization. The SLA provide quality metric verification methods which can be used to test product risk, and it can be used to decrease post-production risks.

B. FINANCIAL MANAGEMENT

One of the most important tasks that a program manager performs is obtaining and retaining funding for the project. Before a project is started a mission need statement (MNS) or a project overview statement (POS) must be approved. The MNS and POS essentially define a problem that needs to be addressed, it describes how the problem will be solved or what the project will consist of, it states why the project is needed, and it details what specific business value or operational advantage it will provide. (Wysocki) This section makes the assumption that management has already approved a MNS or POS, and funding necessary for a detailed project plan has already been received.

The development of SLAs provides valuable information that will assist the program manager in managing the projects finances. SLAs specifically help financial management in determining the scope of the project, identifying business critical processes and functions, they help to allocate costs, they provide justification for service related expenditures, and they coordinate the IT strategy with business strategies.

In the requirements engineering phase of development, stakeholders must determine the scope of the system. Stakeholders need to determine what they need and do not need in the system. As was discussed in Chapter III, the SLA development process provided an additional venue and methodology to explore requirements, it concentrated on business essential non-quality factors that support critical success criteria, and it looks at long term requirements that will affect lifecycle costs.

Once the system has been scoped, and requirements have been generated, it is possible to estimate the costs, schedule and resource requirements to development the system based on function points, KLOC analysis, COCOMO II, or other software estimation techniques. The program manager can then take these more concise estimates back to management to give them a rough idea of the costs associated with the project (estimates early in the project are not as accurate as those made later in the development process). Those costs can then be compared to the expected benefits to determine whether to proceed with the project.

Program managers are typically fighting for funding with other competing interests. Management will fund those projects that it believes will return the greatest

return for the least amount of risk. Management also expects that they are purchasing a quality product. If management is confident that the program manager has conducted a comprehensive analysis of the requirements, has identified critical success areas or factors, has developed a risk management plan, has formulated a plan to closely monitor development and has developed a comprehensive test plan, they are more likely to fund that project over another project that is not as well organized. SLAs provide management with that confidence.

Chapter X outlines research that demonstrates that IT professionals believe that the use of SLAs will improve software quality. Research has shown that quality improvement, although expensive in the short run can produce cost savings over the lifecycle of the product. The same research also demonstrated that quality improvements were most cost effective at the beginning of the project. (Slaughter) However, the marginal return on quality improvement decreases as more effort in that area is applied. As such, program managers need to determine how much to invest in quality improvement. The SLA development process attempts to make a business case (demonstrate how the IT investment supports and advances business practices) for every SLA. As such, many requirements that are 'nice to have' are eliminated or are deferred to another release. The business case allows management to see the effect of funding cuts on specific SLAs, or their return on investment. It also allows the program manager and management to prioritize the SLAs based on business needs. The SLA development process helps to ensure funding is only spent on mission critical requirements.

To gather the information necessary to negotiate or develop SLA thresholds, it is often important to gather measurements on existing systems. It is important to measure actual performance against that which is expected. In many cases stakeholders have unrealistic expectations such as wanting 100 percent reliability. The SLA development process and template SLAs will help to identify those requirements that deviate from industry standards or benchmarked measurements. Program managers cannot waste funds on unrealistic or unsupported requirements. SLAs can be expensive and it is very

important that the quality thresholds specified can be justified (what are the upper and lower threshold boundaries and what affect will they have on the supported business process).

SLAs are also useful in reducing overall lifecycle costs by concentrating on quality at the beginning of development. Quality factors such as maintainability and security can have long term financial implications if either are not incorporated in the requirements or the design. Quantifiable software metrics assist in making good design tradeoffs between development costs and operational costs. This is important when tight development schedules and limited funding could cause contractors to skimp on quality factors such as maintainability, portability or usability. (Boehm 1991) It is also important to remember that in large software systems, the majority of costs occur after the development phase. Unfortunately, few organizations make conscious tradeoffs between development and maintenance costs. (Vigder)

SLAs are also useful in supporting IT accounting where costs are allocated to specific budget centers or stakeholders. Since SLAs are justified based on business case analysis, the services or benefits that the SLA supports can be traced back to the program management effort, the development effort, or to a specific stakeholder requirement (e.g., finance department). The fundamental benefit of IT accounting is that it provides management information on the costs of providing IT services that support the organization's business needs. This information is needed to enable IT and business managers to make decisions that ensure the IT services organization run in a cost-effective manner. (ITIL)

C. QUALITY CONTROL

Program managers are expected to produce high quality products. Unfortunately, there are numerous examples of failed software projects because program managers did not or could not exercise proper quality control. Quality control consists of the actions necessary to certify that desired standards and quality requirements are adhered to during design, implementation and production. (Tricker) In addition quality control consists of

those activities necessary to detect, document, analyze, and correct defects. (Horch) SLAs are a quality control mechanism.

SLAs help the program manager institute a quality control program by identifying business essential quality factors throughout the system's lifecycle, quantifying those factors in measurable terms, defining how and when the quality requirement was going to be verified, and encouraging the contractor to meet quality goals through penalties or incentives.

The development of SLAs helps make those involved with the process more aware of how quality considerations influence design, lifecycle costs, and performance. SLAs also make management and the contractor more aware of quality in general. The penalties/incentives will help to focus stakeholder's attention on quality issues.

D. MAINTENANCE

Software maintenance is the modification of a product after delivery to correct errors, improve performance, or adapt the product to a modified environment. The modification relates to the code as well as the underlying documentation. The object of software maintenance is to modify the product, while preserving its integrity. (Bennett) The program manager must still maintain quality control over the software even after it has been deployed. Configuration control processes and performance monitoring are essential elements in post-production IT management.

Maintaining IT systems is every bit as challenging as developing new systems, however post-production support does not receive the same resources as a system in development. New systems generally receive the funding, support and oversight necessary to develop the system. Once a system is developed, program management is typically turned over to a functional specialist who deploys and maintains the system. Deployed systems do not generally receive the same funding and personnel resource considerations that they deserve. Businesses are constantly trying to divert more funding from support expenditures to new production.

Maintaining systems is especially difficult with older legacy systems. Older systems are often plagued by inconsistent, inadequate, or missing documentation. These

systems also tend to be fragile when it comes to software migration or modifications. These legacy systems are constantly being pressured to adopt the latest technology, architectural mandate, or respond to new customer or market driven enhancements. Additionally, contractors or junior programmers, who may not understand the "big picture" view of the system, often because of their junior status, are assigned to implement the changes to these older systems. (Prouten)

Ensuring the integrity of the original requirements is extremely difficult as a system ages. As personnel with the tacit knowledge of the original system leave the program office and the contractor's team, the need for accurate documentation becomes more important. To maintain the integrity of the original system, all modifications and maintenance actions must be entered into the configuration management system, where they will be submitted to a CRB with the appropriate documentation, the changes will be tracked and controlled, new identification will be issued, and the change release will be carefully managed. Unfortunately, as systems age, it is not uncommon to discover that programmers have violated standards, architectures and procedures in order to make a system operational.

Software maintenance is extremely important because some studies indicate that maintenance costs can account for up to 70 percent of a system's lifecycle costs, (Hulse) and other place the figure at three to four times the initial cost of the system. (Vigder) Additionally, the maintenance philosophies incorporated into the system design influences programmers' ability to quickly and reliably change software. Slow change equates to lost business opportunities. (Bennett) An example of a systems designed for software maintenance is one that contains architecture that are well defined, clearly documented, and promotes design consistency through guidelines and design patterns. (Hulse) The maintenance philosophy can also have a tremendous influence on the total lifecycle costs of a program. Unfortunately, few organizations make conscious tradeoffs between development and maintenance costs. Many systems are delivered without proper documentation and are given to the maintenance centers without the necessary knowledge. This increases the cost of maintenance and reduces the quality of the work. (Vigder)

In the post-production phase, any proposed maintenance changes or changes to requirements still needs to be reviewed by the CRB. Although the composition of the members of the CRB may change as maintenance contractors or personnel replace those that were involved in the development (an ideal situation is when the people performing the development work are also involved in the long term maintenance of the system) the functions that the CRB perform are still essential.

The CRB review the proposed maintenance action and the effects it will have on the operating system, architecture, functionality, service level agreements, documentation and training. The board also discusses the time frames to implement, security of the source code, methods of issuing the update, effect on interfaces, and scheduling server down time to implement changes. The board also reviews the effect that the maintenance action will have on the underlying processes and business logic built into the system.

It is still important to include stakeholders in the CRB as it is difficult to fully understand and analyze such process issues as information flow, division of work, and coordination without including organizational context in the analysis. Organizational context refers to characteristics of relationships between process participants. (Briand)

The CRB also helps to ensure that the test community is involved in the change management process. The changes need to meet specific performance requirements that need to be specified as part of the maintenance package. The changes need to be incorporated into the testing package so when changes are made, the test community will verify that the changes actually meet the specified requirements. In some cases it is difficult to determine the actual status of a program. Many organizations that do not include the test community in the CRB are forced to declare a task complete when the person responsible for the task declares it to be complete. (Vigder) The test community will have processes and metrics in place to determine if a maintenance effort was completed correctly.

The CRB can also be helpful in evaluating the effects that new technologies will have on the system. As business needs change and new technology is introduced, the system may have to undergo dramatic change to incorporate proposed modifications. Often, management proposes the adoption of new technologies without consideration of

what happens when the software has to be changed. For example, object oriented languages were supposed to make maintenance much easier, however, these languages must be designed with care (e.g., controlling inheritance and threads) or their maintenance can be more difficult that traditional languages. (Bennett) The CRB along with contractors can help the program manager scope the maintenance project and what it will take to accomplish in terms of cost and schedule.

Quality control is stressed during the development of software, but it is rarely evaluated after the application goes to production, unless there are major problems. The program manager must constantly monitor the program throughout its lifecycle to measure the effectiveness of the program, quality, and to detect early signs of problems that may require maintenance action. The SEI quality framework lists attributes that may help program managers track and categorize problems. This information can improve overall knowledge about problems within the program, and can be used to determine if maintenance action is warranted. (Kajko-Mattsson)

SLAs can be utilized for the maintenance actions in much the same manner as development efforts. Software quality can be improved in the maintenance phase by utilizing SLAs to ensure the contractor adheres to SLA mandated documentation requirements, specific standards and processes (configuration management process), quality requirements (defects, complexity, security), and performance requirements (throughput, availability, response rate). As most of the program managers in the post-production phase do not have a technical background, template SLAs can help them understand the metrics that should be collected when maintenance action is performed. Although the program manager may need assistance modifying the template SLAs to meet the unique maintenance needs of the system, the major quality areas will be addressed, and the program managers will be more informed.

E. CONTRACT MANAGEMENT

Organizations are becoming more reliant upon IT as a tactical and strategic business tool. IT has provided organizations with the increased computational powers and communications to rapidly process and act on data. The advent of e-business

(business utilizing the Internet) has introduced a new distribution channels for goods and services, increased corporate partnerships, introduced new markets, and has lead to innovations such as just-in-time inventories. IT has also enabled organizations to become flatter, allowing them to respond and adjust to external forces quicker and more effectively. Organizations that can leverage IT better than their competitors will gain a significant competitive advantage.

As technology rapidly advances, these mission essential IT systems are becoming more complex and more difficult to manage internally. Many organizations have discovered that they do not have the necessary IT skills within their organization to develop and/or manage these systems. Rather than hire IT specialists, or invest in training for their staff, they are considering outsourcing their IT work as a strategy. This is especially true for smaller businesses that cannot afford to keep the in-house IT staff necessary to develop, maintain, and monitor IT intensive systems.

Outsourcing is the process of contracting with a service provider to perform a function or functions that used to be performed by the organizations own (in-house) staff. Outsourcing has been a business strategy for a number of years. Organizations are generally more comfortable assigning functionality to in-house staff as it gives them more flexibility, they do not need to contract for the services, in-house staff already knows the organization's policies and procedures, they have greater trust in their own staff, and in many cases they were cheaper than contractors. However, as more specialized skills are needed to develop and maintain IT intensive systems, outsourcing is becoming more advantageous.

The emergence of companies specializing in providing IT services (external service providers (ESP)) have provided a source of IT specialists that can in many cases provide high quality service for lower prices than internal IT organizations can. IT outsourcing is gaining popularity and is increasing in volume worldwide. In many cases IT managers have little choice but to outsource as ESPs provide access to cutting edge technology and skilled staff, they share the project risk, and they allow organizations to concentrate on core competencies, and they can be cheaper. (King, Goth, Greaver, Nelson)

However, outsourcing efforts require additional discipline and management oversight that may not be necessary with in-house development and maintenance. Program managers not only need to be involved in requirements determination, risk assessment, quality management, change management, and test and evaluation, but they must also be involved contract preparation, contractor evaluation, proposal evaluation, contract tracking and oversight, and contractor performance management. The program manager must be an informed buyer. (Feeny) Program managers must develop strategy to deal with ESPs that includes how the program manager will manage the contract relationship, access to proprietary information, chains of command, monitoring policies, dispute resolution procedures, and early termination.

Contract management is one of the program manager's most important tasks. The purpose of contract management is to obtain the services that are defined in the contract and achieve a return (business value) on the investment. (Lewis) A poorly developed and managed contract can quickly lead to performance and fiscal problems. Contractors are profit driven, nothing that they do is altruistic; their stockholders will not allow it. As a result, contractors are looking for every cost cutting measure that they can employ to maximize their profits. While not the majority, there are contractors that will not fully meet requirements (e.g., cutting corners) if they believe they can get away with it. Other contractors will take advantage of vague requirements to deliver a cheaper product that may not meet user expectations. The program manager needs to develop a contract that accurately specifies the requirements (terms and conditions for acceptance of the deliverable); while at the same time holds the contractor accountable. The program manager must also balance the desire to constantly monitor and control the contractor with the reality that a partnering relationship works better than an adversarial one.

1. Contact Preparation

This section will discuss contracting as it applies to outsourcing of IT services, but the same concepts can be used internally between a business entity and the IT department. Contracting for IT services can be very complex, especially when dealing with the government where the Federal Acquisition Regulations (FAR) and Defense Federal Acquisition Regulations (DFAR) must be followed. A detailed discussion on

contracting is outside of the scope of this dissertation; therefore this section will oversimplify the contracting process to emphasize the positive affects that SLAs have on the process.

When contracting for IT services, the organization requesting the services needs to first determine their requirements. Those requirements (including the SLAs) are incorporated into a document called a request for proposal (RFP). The RFP is sent to organizations that the contracting officer believes can perform the work requested. In the government, the RFP is advertised in the Federal Business Operations, (formerly the Commerce Business Daily). Those organizations responding to the RFP or to the Federal Business Operations submit a statement of work (SOW) that describes how they will meet the requirements requested in the RFP. The SOW also includes the organization's estimate on how much it will cost to provide the service, and a schedule that defines how long it will take to start or provide the service. When the contracting officer has received SOWs from the organizations interested in performing the work, proposal evaluation begins. The contracting officer evaluates the SOW for competency (demonstrating an understanding of the domain and contracting procedures), professionalism (responsiveness to RFP), risk, costs, schedule, past performance, and technical proficiency. When a contractor is selected to perform the work, a contract is written, which specifies the requirements, and contract type (e.g., firm fixed price, cost-plus, cost-plus incentive). At this point the contracting officer and the organization negotiate a price and timeline for the service, as well as other terms such as control of intellectual property rights and whether equipment or material will be furnished to the contractor to perform the requested service. When a price is agreed to, the contract is awarded, a contracting officer representative is assigned to manage the contract performance, and work begins.

Throughout the contracting process (i.e., before contract award), the contractors and the organization's contracting officer are meeting and exchanging questions to ensure that the contractor understand the requirements, and in some cases to educate the contracting officer and the program manager about conflicting requirements or technical feasibility. The vendors bidding on the contract want to ensure they perform due diligence so they understand the scope, the work to be accomplished, performance and

quality criteria, the operating environment, what deliverables are expected, schedule constraints, and acceptance criteria. When the vendors feel they understand all of the requirements, they can begin to prepare their SOW that will detail how they will accomplish the work.

The foundations of contract management are laid in the contract itself. The contract should specify agreed levels of service, quantifiable functional and non-functional attributes, incentives, timetables (milestones), measures of performance, communication channels, escalation procedures, change control procedures, and price. (Lewis) Well written contracts also define the authority that each party has to assign, remove or supervise personnel from the contractor's team, intellectual property rights, ownership of the source code, terms and conditions to terminate or modify the contract, use of third party contractors, transfer or purchase of equipment, migration plans, and acceptance critieria. (Chorafas)

SLAs help to form the foundations of the contract because many of the elements of the contract such as escalation procedures, quality thresholds, points of contact, and roles and responsibilities are already incorporated if a template SLA similar to those found in appendix (A) is used. Strong formalized requirements along with performance monitoring can help to improve the working relationship between the vendor and the contractor. Poor contracts lead to friction, which in turn leads to distrust and ultimately results in poor performance. (Chorafas)

A common understanding of the goals of the project and a monitoring system that identifies and resolves problem issues before they affect contract performance creates an environment that is more conducive to forming a good partnership. A good working relationship requires continuous meaningful two-way dialog between the organization and the contractor. SLAs help establish communication by identifying the chain of command, escalation procedures, and identifying the individual(s) that will be monitoring the SLA. In addition, the very process of monitoring the SLA will in many cases open dialog between the monitor and the contractor that may identify problems, or signal that the contractor is meeting or exceeding all requirements.

Contracting for services requires that all stakeholders and the contractor have a clear understanding of the requirements. It requires a great deal of time and effort to craft a contract that accurately describes the deliverables and acceptance criteria. There is a tendency to write ambiguous language into the contract in the hope that as the contract progresses details can be worked out. This is common when there are time pressures forcing the program manager to get the contract signed and get the work started. Unfortunately, unless there is a great working relationship between the organization and the contractor, there will be conflicts when it comes to defining the small details. In many cases contract modifications are needed to better define the requirements, and extra funds will be needed before the contractor will execute those new requirements. Organizations will have little contractual recourse if they disagree with the contractor's interpretation of their ambiguous requirements.

The SLAs development process and template SLAs show organizations the value of writing very detailed requirement specifications for the product. Detailed specifications make it much easier for any organization (in-house or outsourced) to deliver a quality product on time. (McLaughton) Detailed specifications also make it much easier for contractors to put together a bid on the RFP. Precise requirements allow the contractor to make better estimates of the resources (manpower, skills, funding) and time that it will take to complete the project. (Lewis) The more effort that the contractor can put into the bid, the easier it is for the organization to evaluate.

It is not unusual for organizations to bid low (i.e., low ball or buying in) on a RFP to get the contract. Once they get the contract, they send in a team to perform true due diligence to determine what it will cost to actually perform the services specified in the contract. If they underbid the contract, they look for additional work that was assumed, but not implied in the contract, and they look to recoup funds by overcharging on additional requirements that are generated during the development or support effort. Either approach tends to strain the contract relationship. It is important to note that any additional work must be accomplished through a contract modification, where the contractor must demonstrate that there were deficiencies in the RFP, or that new requirements have been generated. When SLAs are included in the contract, contractors

are more likely to take the time to develop good estimates and determine what steps are necessary to accomplish the tasks while reducing their risks, because the financial risks (penalties or incentives) of not doing so can be severe.

SLAs are useful in contracts not only because they concentrate on quality factors, but they also have the ability to penalize the contractor for non-performance without having to resort to termination clauses (In government contracting the term 'penalty' is used to represent the withholding of any incentive payments or bonuses associated with the SLA, the FAR does not allow a fine for nonperformance). Most contracts include termination provisions where a contract can be terminated if the contractor is not abiding by the terms and conditions of the contract (requirements, processes, cost or schedule constraints, personnel turnover). Unfortunately, while it may be advantageous to terminate a contractor for fiscal reasons, it achieves little in terms of fielding the system. As a result some contractors will work at the minimum accepted levels of performance in an effort to gain more profits. To motivate contractors to perform better many contracts include incentives, which are normally based on cost and schedule thresholds. Incentives are normally based on passing milestone reviews, with the assumption that the reviews will determine whether functional requirements have been met or not.

SLAs support standard contracts by providing incentives or penalties for achieving or not achieving quality thresholds throughout the lifecycle, not just at the milestones (In government contracting the SLAs provide the quality threshold and the associated penalties or incentives, but the contract itself, which will refer to the SLAs, provides the incentives). This gives the program manager more options. In most contracts, if a contractor has met functional requirements on time and on budget, but its configuration management system is poorly maintained, there are few options that the program manager has other than writing a poor evaluation/recommendation to resolve the problem. Termination clauses generally do not address quality issues, which have a lower priority than functional requirements, cost, and schedule. If SLAs are used, incentive pay can be withheld for the reporting period agreed to in the SLA (monthly or quarterly) or the contractor can be fined until the configuration system meets the quality threshold. If the problem persists, the program manager has the option of terminating the contract

(write termination clauses into the SLAs for persistent failure to achieve thresholds), or if the program manager sees improvement, the incentive pay can continue to be withheld until thresholds are met.

In outsourcing contracts, quality is best achieved by comprehensive and detailed requirements specifications coupled with well defined SLAs with built-in penalties should service levels go awry. (Chorafas, Baron) The SLAs help to reduce overall contract risk by monitoring quality throughout the lifecycle. Most SLAs measurement periods are over a monthly or quarterly time period. Accordingly, problems with meeting quality thresholds are identified long before a milestone review. This contract monitoring allows the program manager to quickly take action to resolve the problem, and if necessary to terminate the contract before too much time and money is spent.

2. Proposal Evaluation

Once SOWs are received from contractors interested in performing the requested services the organization must develop a methodology to select the contractor that can best meet their requirements. The criteria used to evaluate proposals should be determined before the RFP is completed to ensure that the RFP effectively communicates all of the areas that need to be evaluated. The evaluation criteria must be included in the RFP. In most cases the evaluation consists of a balance scorecard type of approach where weights are attached to specific attributes such as reputation, price, schedule, risk, and processes.

The process of selecting a business partner should be well thought out. A good partnership can provide benefits to both organizations; however, a poor relationship can jeopardize the project, alienate customers, anger stockholders, and damage both organizations' reputation.

In some cases a pre-qualification can be accomplished to limit the amount of applicants. Pre-qualification audits or screens are done to ensure that the organization is not wasting its time evaluating a contractor that does not have the capability to satisfy the conditions of the contract. (Roberts) Pre-qualification audits review the SOWs to evaluate the number of staff and their skill sets, the financial condition of the contractor,

pending lawsuits, the reputation of the contractor (check references), CMM ratings if applicable, the type of work (technical level and complexity) the contractor has done in the past.

SLAs aid organizations in the pre-qualification of applications. SLAs contain quantifiable quality requirements along with a methodology to confirm whether the requirements have been met. The detail of the requirements along with non-performance penalties will generally discourage all but the most serious contractors. The SLAs tend to limit the proposal to only those that are capable of providing a quality product or service.

When the pre-qualification has been completed the remaining proposals are reviewed. A more detailed analysis is conducted of the proposals and the contractors. Although many factors are scored (balance scorecard), the selection criteria can be grouped into seven main categories. The categories and the way they are scored should be aligned to the underlying business processes that the IT system supports, and the overall business goals of the organization. The first category evaluates a contractor's quality control and quality management processes. The second category looks at the technical competency of the contractor in terms of employee skills, tools, training programs, innovation, and past performance. The third category analyzes the contractor resource management practices in terms of employee management (employee turn-over, pay, training opportunities) and knowledge management (how is tacit knowledge captured, how is information collected and shared). Determining the financial strength of the contractor is the forth category. The fifth category determines whether there is a good cultural fit between the organization and the contractor (e.g., a contractor may operate in an environment that has to rapidly respond to the business environment, the contractor will have to be have quick, flexible processes to accommodate that need). The sixth category evaluates the contractor's program management processes, such as configuration control and change management. The last category is the costs of the project and projected costs over the lifecycle of the project. (Roberts)

SLAs are also helpful when scoring the proposals. The quality factors represented in the SLAs represent those areas that stakeholders felt were essential to achieving a quality deliverable. As a result, the quality factors identified in the SLAs

should be scored higher than other non-essential factors such as the contractor's administrative support. In addition SLAs make it easier to focus part of the assessment on the contractor's ability to meet the quality thresholds specified in the SLAs. If maintainability is a major concern to the organization, the assessment can evaluate the configuration control system that the contractor used on past projects. The SLAs allow the assessment team to focus on specific areas rather than conducting a general overview of the contractor's processes and past work.

3. Contract Oversight

After the proposal evaluation is completed and a contractor is selected the details of the contract are negotiated. When both parties sign the contract, the process of contract oversight starts. The main purpose of contract oversight is to ensure that both parties are fulfilling their contractual obligations. (Hill) SLAs were developed in part to provide contract oversight by monitoring the quality factors specified in the contract. In this dissertation contract oversight is broken into maintaining a good relationship between the parties, and ensuring the contractor is adhering to the terms and conditions of the contract.

There are a couple of different types of contractor-organization relationships. A partnership is a formal business relationship that is established to achieve common business objectives. Partnerships are usually long term and are characterized by a close working relationship where the contractor is an active team member. In partnerships the organization and the contractor have a vested interest in the success of the project. (Hill) An affiliation is also a formal business relationship where pre-qualified contractors are engaged, as their services are needed. Examples of an affiliation are buyer purchase agreements (BPAs), where the service and price have already been negotiated, and a contract is executed only when the service is needed. Another formal relationship is the project specific relationships where the contractor is needed on a specific project. This type of relationship is very common and it includes RFPs, SOWs and a selection process. The last type of relationship is a service provider relationship where the contract may be formal or informal. An example of this type of relationship is the local server hardware maintenance professional who has been pre-approved to do preventive maintenance work

(i.e., run diagnostics, vacuum dust) for the organization. When the maintenance man's services are needed, he is called. The maintenance man provides a quick estimate of the cost of the job, and if the price is acceptable, the organization will contract for the services (in many cases an account already exists). (Hill) In each case a good working relationship is beneficial to both parties.

There is a common misconception that SLAs can cause an adversarial relationship between an organization and a contractor as a result of penalties for noncompliance. However, many contractors like SLAs because they define the services that must be performed in detail, they provide the quality thresholds that must be met, and they state the means by which those services will be measured. The detail provided in the SLAs helps to prevent much of the ambiguity that causes disagreements. Both parties agree to SLAs; and if a contractor does not meet requirements, then they understand the repercussions, because they also understand the effect that not meeting those requirements has on the organization. Contractors expect to be penalized for poor performance; problems arise when there are differing interpretations as to the services being provided, and their associated performance requirements. Specifying the methodology to verify compliancy also eliminates many of the arguments that may occur. As was discussed previously, depending upon the organization-contractor relationship, the maturity of the technology, or how well requirements are understood, it may be better to structure the SLAs as incentives instead of penalties.

Managing the relationship between a vendor and an organization is a difficult but extremely important task. Both parties need to understand the motivations of the other party to be successful. Contractors are motivated by profit, but they must price their services to be competitive with other contractors and the internal IT shop within the organization. Contractors try to not only win the contract, but they want to establish a good long term working relationships to gain more work and generate additional profits. Organizations want a system that performs to specifications, so the system can enable business processes that will allow them to generate profit. The solicitation process is the means that the organization uses to ensure they are not paying too much for the service (competition will lower the price of the service), and the contract is the process that they

use to ensure they will receive the functionality and quality that they desire. Organizations must also understand that if the contractor is not making profits, the risk of default or non-performance on the contract increases significantly. SLAs tie the vendor's most important concern, profits, with the program manager's most important concerns, performance and quality. (Agarwal)

The program management office needs to develop procedures and processes to manage the contractors. The program manager needs to determine the type of information that the contractor needs access to, whether the contractor is included in daily meetings, whether they are managed at a distance, how information will be shared (e-mail, meeting minutes, central repository), the chain of command, security clearances, and the degree of freedom that the contractor has to develop solutions or to resolve situations. If the SLAs include end-to-end components or if the system is a part of a system or systems, the program manager may have to manage multiple development and maintenance contracts with many different contractors. The program manager will have to determine how to manage the various contractors and their interactions (i.e., are contractors allowed to communicate among themselves, or do they have to communicate through the program management office).

SLAs provide information that helps both parties manage their relationship better. SLAs identify the individual who is responsible for managing the SLA. Depending upon the complexity of the system, manpower availability, and the criticality of the system the SLA will assign an individual to act as a contract monitor who is responsible for verifying that quality thresholds have been met, but an addition individual may be needed to act as a contract facilitator who would be responsible for working with the contractor to resolve day-to-day issues relating to the SLA. (Currie) In smaller projects, the same individual will perform both functions. Although one individual may be responsible for multiple SLAs, it is helpful to specify the specific point of contact for each SLA as it helps to build and maintain the organization-contractor partnership.

Conflict is a normal part of the development process, requirements are not always known well enough to specify in exacting detail, systems are complex, and the business environment is dynamic. The ability of both parties to resolve these disputes amicably

will determine the strength of the working relationship. SLAs help to provide some structure by designating responsibilities for various tasks as they relate to the SLA. This definition of roles and responsibilities provides greater clarity and better defines the working relationship. The SLAs also state assumptions that were used to build the SLAs, which may also resolve possible disputes before they occur.

4. Contractor Performance Management

SLAs help to manage the contractor by defining the quality factors and metrics that must be met, they define how the metrics will be collected, they increase communication between the contractor and the program management team, and they define roles and responsibilities of both parties.

The key to contractor performance management is oversight. The program manager is responsible for ensuring that the contractor is complying with the terms and conditions of the contract. The program manager must also verify that any deliverables meet stated requirements. It is very important that quality control measures are in place to inspect and verify the contractor's product at each milestone. (Hill) SLAs explicitly state the quality factors that an organization expects in the end product. SLAs also explicitly state the metrics and the collection mechanisms that will be utilized to verify that the quality requirements have been met. SLAs also establish a monitoring process to verify compliance with quality requirements. As such, any deviations from the organization's expectations can be quickly resolved before they become major problems. Additionally, monitoring provides information to utilize in forecast analysis.

When the contractor was preparing the solicitation in response to the RFP, the contractor had an opportunity to challenge or question any of the SLAs. If the contractor decided to bid on the contract, then they agreed to abide by the SLA. SLAs establish a clear understanding of the product quality, process quality, production quality and post-production quality expectations. Although SLAs place constraints on the behavior of contractors, numerous contractors interviewed have indicated that they favor contracts that clearly articulate expectations as it resolves may of the conflicts that normally occur over interpretation of requirements.

During the solicitation process, or in some cases if the contractor participated or lead the requirements engineering process, the SLAs generated meaningful communication between the contractor and the organization. The SLAs not only introduce quality requirements at the beginning of the development cycle, they also generate discussion on standards, testing, monitoring, design, critical business processes, change management, quality models, and quality control. These discussions hopefully, improved the SLAs, established common frames of reference, and established a good working relationship between the parties. The reports generated as a result of the SLAs also help establish communication between the contractor, program manager, end users, and upper management.

Many contracts drafted by lawyers include long, tortuous statements full of legalese and cross references that are difficult to understand. (Nellore) Lawyers do not draft SLAs, they are written by end users, management, IT personnel, and business process owners. Lawyers should review SLAs to protect the organization, but they need to be understandable by all parties involved. The ease of reading makes SLAs more effective in communicating requirements than some contracts.

Contractor performance management is more than monitoring quality metrics and assigning blame if they are not met. Contractor performance management also needs to monitor the relationships between all parties. Blaming the contractor for quality problems does not solve the problem. If relations between the organization and the contractor reach a point where both sides are blaming the other for problems, then both parties loose.

Although both parties may not have the same objectives or policies, and both have constraints (internal and external) that influence their behavior, SLAs can be used to influence both parties to take appropriate actions to come to the mutually accepted behavior as agreed upon in the contract. (Milosevic) SLAs specify the roles and responsibilities of both parties, they specify the assumptions, they specify quality expectations, and both parties agree them upon. SLAs also specify procedures for dispute resolution, so issues can be resolved quickly.

F. CUSTOMER SATISFACTION

Another important task that the program manager has is ensuring that all major stakeholders are pleased with the delivered product. A program can meet cost, schedule and performance parameters, but if the stakeholders are not pleased with the product, the perception will be that the project failed. Customer satisfaction is an important role for the program manager. The program manager must ensure that the delivered product is acceptable to the stakeholders; however, the program manager must also ensure that once the product is delivered, that it is properly supported through the use of SLAs.

The process of developing the SLAs helps the program manager by establishing buy-in from the major stakeholders. Representatives from the major stakeholders are able to participate in the development process, and they determined the quality requirements and quality metrics that they felt best support the business critical processes. They also have the ability to state their own expectations and make a case for quality factors that they feel are important. When those stakeholders return to the positions they left, they are generally advocates for the program manager and the SLAs because they helped develop them.

The program manager can utilize the SLAs to set customer expectations. The SLAs define the quality factors and the quality thresholds that the user can expect. The SLAs also demonstrate that the program manager has an aggressive plan in place to monitor performance and penalize the contractor if quality thresholds are not met. SLAs also help institutionalize the change review board, which helps to inform users of approved changes to the system. The SLAs help to prevent expectation creep, a situation where users constantly want better and faster performance. The program manager can easily point to the SLAs and declare that despite the user's concerns, the stakeholders have determined that the current quality levels are sufficient to support the critical business processes.

Program managers need to monitor customer satisfaction to ensure that the services are meeting end user needs. A survey is one method of measuring whether end users are satisfied with the services that a contractor is providing. SLAs can be written such that the contractor needs to achieve a certain score (90% satisfactory or above) to

meet a quality threshold. Before SLAs are developed for surveys, the program manager should assess the environment to ensure that the contractor will have a chance of meeting the quality thresholds. If the internal IT department lost jobs because of work outsourced to the contractor, end users may have a hostile attitude toward the contractor. In that case, the program manager may want to wait until attitudes towards the contractor have softened.

An important part of customer service is monitoring the performance of the system to ensure that it is supporting the critical business processes in a manner acceptable to the customer. In the deployment or post-production phase of a system's lifecycle the host provider (whether those functions are outsourced or kept in-house) must perform certain services to keep the system operational. Service-level management (SLM) is the proactive methodology used to ensure that adequate levels of service are provided to all users in accordance with business priorities. (Sturm) SLM involves monitoring, reporting, modifying, and improving the quality of the services being provided to an organization. SLAs are a part of SLM in that they define the services to be performed, and the levels of service expected.

Some of the areas of SLM include availability management, quality of service, and resource management. An integral part of maintaining an availability threshold is the constant monitoring of each of the hardware and software components that comprise the system's infrastructure. Components that are not performing as expected, should be examined and action should be taken to resolve any problems. This may require additional monitoring, trend analysis, or changing to another component from another vendor.

When measuring the network infrastructure performance, traffic behavior needs to be evaluated with respect to four characteristics: importance, time sensitivity, size, and jitter. For applications that are critical to the success of the organization, efforts need to be taken to protect its performance. This may mean allocating bandwidth specifically for the application, or prioritizing those packets in a QOS scenario. Application traffic that is time sensitive, interactive, or subject to latency problems will also need prioritization (e.g., telnet or Oracle). Applications that have network traffic that expand to meet the

amount of bandwidth available, or produce large surges of packets (e.g., FTP, streaming video, *.jpg files) can have a negative impact on other applications. Bandwidth hungry applications can deprive higher priority traffic of necessary bandwidth. Streaming applications need a minimum bits-per-second rate to deliver acceptable performance. The bandwidth needed to support these types of applications (e.g., VoIP, Real Audio) need to be balanced against available bandwidth, the business value of the application, and the needs of other applications. (Packeteer, May 2002)

Capacity management provides the necessary information on current and planned resource utilization of individual components to enable organizations to determine which components to upgrade, when to upgrade and how much the upgrade will cost. (ITIL)

Service capacity management needs to monitor, analyze, tune, and report on service performance, establish baselines and profiles of use of services, and manage demand for services. (ITIL) It is important that a good baseline be established so the service provider understands the resources and capability requirements of the application.

Capacity management helps mitigate risks associated with resource requirements. Proper planning ensures that an application will have the resources necessary to execute all functionality to specifications. Capacity management is also involved in analyzing the resource needs resulting from any application modifications approved by the change review board. In the host environment, new applications, or modifications to existing applications can affect the resources (e.g., infrastructure) used by other applications. Accurately predicting resource needs of the new application, in addition to information collected on the usage of other applications will ensure that there are enough resources for all of the application, or identify the need for additional resources. (ITIL)

G. SUMMARY

The use of SLAs helps the program manager with many of the tasks necessary to managing complex IT systems. The development of SLAs improves the communication between the stakeholders, management, and the contractor. Increased communications helps to improve relationships, identify risk areas, better understand the requirements, and it leads to better problem resolution. The monitoring processes resulting from the

SLAs help the program manager monitor performance and contractor compliancy. SLAs need to define the quality requirements in great detail to ensure that all parties understand the quality expectations for the system. Well defined quality requirements reduce the possibility of conflict due to misinterpretations of requirements, and helps to set user expectations regarding performance. SLAs can also be utilized to entice the contractor to take the necessary measures to ensure that their quality control measures are in place and are accurate.

X. RESEARCH METHODOLOGY

A. PHILOSOPHICAL APPROACHES

The objective of the questionnaire is to gather evidence to support the hypothesis that service level agreements can increase software quality and management of IT intensive systems. If the hypothesis is supported, the results of the questionnaire can be predicted. If the outcome of the questionnaire is similar to predictions, then the hypothesis is supported. The questionnaire is designed to demonstrate the causality between the hypothesis and expected results. (Xia)

There has been a great deal of debate on research methodology within software engineering field. Much of the debate centers on the various philosophical approaches to ontology (the nature of being) and epistemology (the theory of knowledge). Those beliefs drive the methodology in conducting research and engaging in problem solving. The various philosophical approaches can be grouped into four distinct groups. (Reeves)

The first group is the analytic-empirical-positivist-quantitative group. This group is most often identified with mathematicians and physicists. This group believes that the world is deterministic, or it is operated by the laws of cause and effect. Research methodology associated with this group is generally highly structured and is centered on laboratory experiments. This group believes in empiricism, or the idea that observations and measurements are the core of the scientific endeavor. (Trochim) Problems are decomposed into elements, variables, covariants, attributes, and values. Tests are conducted under controlled conditions to not only establish a repeatable test, but also to systematically alter the variables, observe the phenomena, and measure outcomes against predictions. This group only tests what they can measure and observe.

The second group is the constructivist-hermeneutic-interpretivist-qualitative group. This group does not subscribe to the detached, objective, nomothetic approach to research. This group views the nature of reality differently. This group believes that reality can only be defined by multiple perspectives, and that factors such as culture, sex, context, and emotion influence individual perspective. They believe that laboratory experiments are a poor substitute for testing ideas in organizational contexts using real

practitioners. (Moody). They view information technology as an applied science instead of a pure science. Scientific rigor does not apply well to applied science. This group tends to borrow many of the research methods used in anthropology and sociology. (Travis) This group utilizes focus groups, interviews, and case studies. These are the same techniques used in requirements elicitation. (Pressman, Nuseibeh, Galliers)

The third group is not as representative as the other two, but they bear mentioning. They are the critical theory-neomarxist-postmodern-praxis group. This group believes that all assumptions must be challenged. This group is essentially anti-establishment, believing that there are hidden agendas and contradictions in most research. They are critical of our ability to know reality with certainty. (Trochim) They challenge the underlying cultural, legal, scientific assumptions that form the basis of reality. (Reeves) For example, Einstein postulated that light (c) has a constant speed regardless of the frame of reference. Numerous experiments have confirmed his postulation, however recent work by Montgomery and Dolphin are challenging that postulation. Their research has indicated that the speed of light decreased over time, thus the atomic clock is decreasing with respect to dynamic time. (Montgomery)

The last group is the eclectic-mixed methods-pragmatic group. This group is not averse to using techniques of the other three groups to collect data and solve problems. The approach used depends upon the problem to be solved (i.e., hypothesis), the context in which it resides, and the purpose of the research. (Travis, Moody) This group tends to be more practical and they are not as philosophically driven as the other groups. They recognize the weaknesses of the various methodologies and try to construct an approach that maximizes the value of the information gained in relation to the objectives of the research.

B. APPLYING VARIOUS METHODOLOGIES

Developing a pure positivist approach to supporting the hypothesis in this dissertation is difficult. In scientific rigor, all variables that affect the end result (i.e., quality software and post-production support) must be identified. To ascertain that results are only caused by the hypothesis, and not other conditions, other irrelevant

factors must be controlled and kept constant to eliminate their influence. (Xia) It is therefore necessary to identify all of the factors that lead to quality software and support. This approach makes the assumption that the concepts of quality and support and their associated properties can be defined in measurable terms. It is possible to use the quality measures derived by McCall, Hewlett-Packard, and ISO 9126, but these measures are indirect measurements of quality, and are often subjective.

It is possible to rephrase the hypothesis in terms that are more quantitative, but that does not make defining the terminology any easier. For example, if the hypothesis stated that SLAs could reduce coding errors, the end result is still not clearly defined, and it does not address the underlying theory. The hypothesis does not state whether the errors that will be reduced are in the development, coding, or maintenance stage of the application's lifecycle. The hypothesis also fails to explain how coding errors are reduced. To properly test the hypothesis, all of the factors contributing to coding errors would have to be explicitly defined. Establishing a control group to test the hypothesis will be difficult, when variable factors such as education, experience, code complexity, fatigue, and time pressures contribute to coding errors. Researchers ascribing to the positivist beliefs need to be careful to avoid the pitfall of focusing only on problems that can be researched (using scientific rigor) rather than those problems that should be researched (i.e., provide practical knowledge). (Moody)

Interpretivists claim that software cannot function in isolation from the system in which it is embedded, and a systems view necessitates evaluation of human factors. They believe that many software methodologies, heuristics, and guidelines are dependent not upon pure scientific research (i.e., positivist approach), but upon human cognition (Xia), social action, and even the human body (Mingers). An interpretivist approach to testing the hypothesis would consist of group discussions or individual interviews to determine people opinions regarding the hypothesis. The core of interpretivist research is the need to understand the relationship between an individual's behavior and that individual's mental state of preparedness to act in a predetermined way. (Smith) The researcher starts with an existing (theoretical) knowledge of the topic under investigation. Through a

process of interviews, the researcher gathers new evidence and compares the results against what is already known about the phenomenon under investigation. (Smith)

It is possible to test portions of this hypothesis using an interpretivist approach, but testing the entire hypothesis will be extremely time consuming and will also be very difficult. Interpretivist studies are designed more towards the development of concepts, generation of new theory, examining relations between attitudes and behavior, mapping an individual's overall range of behavior and attitudes, and collecting a rich amount of insight into an issue. (Smith) In qualitative research, the goal is to establish a match between an aggregation of subject's view of reality and the reality that the researcher has. Results obtained from qualitative data are generally not used to support theoretical propositions. This is due in part to the argument that social sciences (e.g., anthropology, sociology) cannot explain events by cause and effect, because they cannot capture all of the contributing factors.

If the hypothesis was that software developers would be more likely to spent the requisite time and effort to reducing coding errors if SLAs with strong incentive or penalties were utilized, then individual interviews or focus groups could discuss SLAs, and whether the incentives or disincentives motivated them to produce faster and better code. Researchers could ask questions such as, "If you were fined for each error you produced, would you concentrate more on reducing errors?' Another question might be "In your company are you evaluated by quality or quantity of code produced?" By comparing the results of the research against predicted outcomes, the researcher could determine if the evidence collected supported their view of reality.

The eclectic group believed that it was possible to combine positivist and interpretivist methodologies to derive a richer solution set. Limiting research to one type of methodology offers a limited perspective. The best hope of achieving objectivity in research is to triangulate across many different perspectives and approaches. (Trochim) Software engineering is not computer science; it involves a great deal of human interaction and subjectivity. As such neither positivist nor interpretivist approaches can provide an overall solution. Rather than concentrate on a specific methodology to use, it

is more important to determine what critically, theoretical, and practically informed mix of methodologies best deals with the problem to be solved. (Clarke)

C. DISSERTATION METHODOLOGY

The research conducted in this dissertation will combine both the interpretivist and positivist approach. When this approach is used, researchers gather information, opinions, and attitudes concerning a particular topic by which to form propositions. When the interviews reach a point of sampling saturation (i.e., the point where new interviews fail to reveal any new insights), the information is compared to predictions. New insights are collected and the original proposition is supported, or amended to reflect the new information. At this point the positivists can start gathering statistical data such as determining the frequency at which the issues, ideas, or insights occur. The statistical data will offer additional data to apply towards a new or already existing hypothesis. Utilizing positivist approaches towards information obtained through interpretivist research in not unique (Sarker, Kumar, Smith) and can be used very effectively.

At the beginning of a study it is helpful to utilize qualitatitive research methodology to establish an aggregate of people's 'frame of reference' toward a given topic. When issues involve subjective interpretation, it is recommended that researchers only go directly to quantitative methodology if they fully understand their subject's view of reality. Otherwise obtaining qualitative data first is the preferred methodology. Determining the proper mix of qualitative and quantitative research is dependent upon the problem to be solved, and it is likely that one methodology will have more weight than the other. (Smith)

The research conducted in this dissertation will utilize both positivist and interpretivist approaches. The research will appear to be more quantitative than qualitative, but only because much of the qualitative information was distilled into the information presented in the questionnaire.

The qualitative portion of the research consists of a combination of top-down and bottom up approaches. In a top-down approach to qualitative research, the interviewer

begins the research with a particular view of reality and the research is gathered to support this view. This approach has merit if the researcher's theory is formulated in solid normative evidence. It also adds a form of structure and discipline to the subsequent analysis. (Smith) At the other end of the spectrum is the bottom-up approach. The bottom-up approach is where the researcher has no preconceived notions and during the course of interviews and analysis, the formulation of a proposition is created. Most qualitative research uses a combination of both approaches. The researcher generally has a rough concept of reality, and uses the information collected from the research to compare against that view.

The questionnaire in this dissertation started with an informal qualitative analysis among a number of colleagues. In a series of meetings on server consolidation the issue of post-production support was discussed. After reviewing numerous government and commercial contracts for host support, it was determined that none of the contracts reviewed provided the support required. A new contract needed to be developed. While writing the SOW for post-production support, and the accompanying SLAs, interviews were conducted with system administrators, information assurance professionals, program managers, software developers, database administrators, and commercial external service providers (ESP). In addition to the interviews over 100 articles and books were reviewed. The results of these informal interviews and the literary search formulated the bottom-up approach that generated the starting view of reality.

Once the central theme was developed, an article written by Charles Mann initiated the top-down analysis portion of the research. His article detailed a number of reasons that software quality was lacking, and he proposed legislating software quality as a solution to the situation. His solution was to hold software developers accountable for faulty code. On the magazine's web site, readers were able to post comments concerning the article on a bulletin board. Over 100 people responded to his article. The information from that sample group in addition to those of Mann and other articles detailing poor software quality provided additional information to support the starting view of reality.

To take advantage of the information gained in the previous qualitative research, the questionnaire was designed to both validate the information previously collected (qualitative part), and provide a measurement of the strength of the aggregate opinion (quantitative part).

D. QUESTIONNAIRE

Interviews traditionally have a moderator that guides the discussion in order to obtain the information being sought. When the issues are complex, it is often necessary to provide the appropriate amount of education to ensure that the subjects are knowledgeable enough about the topic to make informed decisions. The moderator must ensure that every group or individual is given the appropriate amount of information, and that all relevant topics are discussed. Additionally, any bias needs to be presented to all participants in the same manner. This is extremely difficult when more than one moderator is used. It is even difficult when multiple sessions are conducted with the same moderator. The approach used in this research is to provide all of the information needed to form opinions explicitly in the questionnaire. This ensures that all participants are presented with the same information, and that any bias that is introduced is presented to all participants.

To generate thought on the subject, opinions generated from the earlier ad hoc qualitative analysis in addition to information derived from literature review is presented in the first section of the questionnaire. The section discusses Charles Mann's article, along with numerous opinions on the issue of software quality. The opinions were from multiple sources and represent many different perspectives. Also in this section is a discussion on the merits of SLAs.

To further illustrate the concepts in a real-world scenario, the second section of the questionnaire is a case study on how SLAs are developed, along with an example of a SLA for availability. The case study provides additional information and allows the subject to apply the lessons learned in the first section to a case study. The second section presents a different perspective from the first section, and also generates thoughts, opinions, and emotions on the subject.

The last section consists of the questionnaire itself. Many of the statements were based on the opinions already gathered with the prior qualitative analysis. The statements were designed to provide a more formalistic validation of information previously collected. Although this approach does not incorporate a mediator, the subject is guided through the discussions by the first two parts of the questionnaire, and the last part allows the subject to express opinions concerning the topic. In addition, the questionnaire includes some open-ended questions in which the subjects are free to express their opinion in their own words, from their own perspective.

The three parts of the questionnaire were formulated from the information obtained during the previous qualitative analysis. The subsequent quantitative analysis will concentrate on how strongly subjects feel about various aspects of SLAs and their ability to improve the quality and management of software intensive systems.

The quantitative phase of the research consists of a number of statements (representing the common themes from the qualitative analysis) and an accompanying 5-point Likert scale. The questionnaire begins with demographic data to provide some possible insight to the analysis. Statements 4 through 29 utilized a bipolar Likert scale that ranged from strongly disagree (1) to strongly agree (5). The Likert scale also incorporates a neutral response (3).

The sample was not random in that IT professionals from both the government and industry were asked to fill out the survey. The topic was considered too complex for a random sample of individuals. The IT professionals were also sought for their practical experience in dealing with SLAs, ESPs, software development, program management, and post-production support.

A great deal of effort was expended trying to balance information presentation with the amount of time a respondent would spend on the questionnaire. If the questionnaire is too long, few people will be willing to exert the time or effort to complete the questionnaire. If the questionnaire is too small, the respondent does not have enough information to form an educated opinion.

Good surveys will contain some catch questions or statements. These statements are closely related to as a previous question. The respondent should answer the same

way to both questions. If the respondent answers differently to both questions it may be an indication that the respondent was simply completing the questionnaire without much thought. This survey contains two such questions.

The questionnaire was loaded on a web page, and the URL was e-mailed to numerous IT professionals soliciting their responses. The questionnaire consisted of four web pages. The first page explained the purpose of the questionnaire, and instructions. The second and third pages correspond to the sections on software quality and the case study. The fourth page was the questionnaire itself. When the respondent accessed the fourth page to provide input to the statements, they give their permission to utilize the information they provided in the research. Appendix (B) contains the actual questionnaire.

E. RESULTS

Results from the questionnaire were captured in an access database. The results were then converted to an excel spreadsheet and statistical information was generated and the results are displayed in appendix (C). A Likert scale was used on the questionnaire to determine the degree to which a respondent agreed, disagreed or was neutral on a statement. Using standard statistical analysis on Likert scale responses can be problematic.

In the questionnaire, responses ranged from 1 to 5. All responses were discrete vice continuous. As such using measures such as a mean (average of all values in the sample) and standard deviation (variability of observed values from the mean) can lead to inference problems. For example, what does a mean of 2.5 infer? Is the difference between strongly agree and agree the same as neutral and agree? Additional information was needed to reinforce the results of the mean and standard deviation. As such measures such as median (the middle value when observations are ordered from smallest to largest) and mode (the value that occurs most frequently in the sample) have also been calculated.

Appendix (C) lists the individual questions, the mean, mode, median, standard deviation as well as a bar chart to visually display the results in percentages. Appendix (C) also lists the T-value and the P-value to determine if the results were significantly significant.

The null hypothesis (H_o) was that $\mu = 3$, or a neutral response. The alternative hypothesis (H_a) was that $\mu \neq 3$. Since the values above and below the mean both have meaning (agree and disagree), a two-tailed test was conducted. Given a sample of 43 responses, a Z value could be used (central limit theorem states that a normal distribution curve can be used with a population over 30), but a T-value would give better results given that the population was not much greater than 30. The significance level (α) is .05. So the null hypothesis would be rejected if $t \geq t_{\alpha/2,n-1}$ or $t \leq - t_{\alpha/2,n-1}$. The value of $t_{\alpha/2,n-1}$ is 2.021 using a population of 41. The extrapolated value for a population of 43 was 2.023. This means that the probability of a type I error (H_o is rejected when it is true) is α or 5 percent.

P-values were also calculated to give a better understanding of where the H_a value would be rejected. Simply comparing the calculated t value against 2.023 forces the reader to accept the significance value of .05. The P-value is the smallest level of significance at which H_o would be rejected. Once the P-value has been determined it can be compared against whatever specified level of significance an individual desires. If $P \leq \alpha$ the H_o should be rejected at level α. (Devore)

F. INTERPRETATION OF RESULTS

The first analysis was to evaluate the responses on the catch questions. These are questions that are closely associated with one another. Questions 18 and 22 both concerned program management. There were only three responses where there was a 2 Likert scale difference. Overall the means of the two questions were the same. Questions 9 and 23 dealt with the affect of SLAs on software quality in the development stage. There were two responses where there was a difference of 2 Likert scales. The difference in the means of the two questions was .0148. A t-test on the means of the two groups of samples showed that the differences between the means were not significant.

Additionally some of the questions that the author predicted a response of agree or strongly agree were intentionally worded to be negative, so the respondent would be expected to answer with a response of strongly disagree, or disagree. The respondents did actually respond with a mean towards 2 on those statements with negative wording. This meant that the respondents were actually reading the questions and were not randomly selecting answers.

Statistics on the respondents indicated that the majority had over 6 years of experience in IT. The respondents were well represented in management (58.1 percent) and IT implementers (41.9 percent), and almost half had more than 1 year of experience working with SLAs.

Questions 4 through 29 only had 3 questions that were not statistically different from a mean of 3 or a response of neutral using the T-value test. The H_o could not be rejected on the questions of whether respondents were satisfied with the quality of software they use, and whether a lack of in-house skills would prevent the development of SLAs. Similarly the question of whether it was too difficult to enforce penalty clauses was too difficult also could not be rejected.

Respondents agreed that SLAs would improve software quality throughout its lifecycle. Results strongly indicated that respondents felt that SLAs could improve software quality in the development and post-production phase. However, in the comments column some of the respondents felt that SLAs must be backed up my managerial commitment to be affective. They also felt that SLAs were not a silver bullet, and must be used in conjunction with other quality initiatives. Along this same vain respondents felt that SLAs could not resolve the quality issue associated with management rushing software to market. The results also indicated that there was neutral to mild agreement that software quality in the software they were currently utilizing was acceptable.

Overall, respondents felt that SLAs would improve software program management. Results indicated significant agreement that SLAs would improve program management through configuration management, change management, managing user expectations, focusing on key performance issues, source selection, and ensuring

underlying business processes were supported. Comments indicated that respondents felt that SLAs could contribute to software lifecycle management, but the level of success depended upon management's commitment to those SLAs.

Respondents also felt that SLAs assisted in the development of requirements. Results indicated that subjects felt that the SLA development process not only facilitates the involvement of end users, but in doing so it also helps to manage the end user's expectations. Respondents indicated that they believed that the team development concept helps to identify those quantitative metrics that are critical to the success of the underlying business process. The survey also indicated that subjects felt SLAs could help inject quality and security into the early parts of the development process.

The respondents approved of the format of the SLA presented in the survey. Results indicated that they felt that the format was easy to understand and clearly defined the services and the methodology to measure whether a requirement met the specified threshold levels.

People taking the survey believed that the work required to generate the SLAs were worth the effort. They also felt that developing SLAs were not too difficult for their organization. The two of the questions that had no significant deviation from a neutral response were on whether the skill sets to develop SLAs existed in their organization and whether penalty clauses were too difficult to enforce.

G. RESEARCH USING HOSTING SLAS

The SOW and SLAs in Appendix (A) were developed to determine in a practical business environment whether SLAs could assist program managers maintain quality in their post-production applications. Appendix (A) contains SLAs that have been developed for hosting services for the NAVSUP claimancy. The intent of the SLAs were to demonstrate the potential to utilize SLAs to manage information-intensive systems and inject software quality in the post-production environment.

The NAVSUP claimancy, like most, has been hit by fiscal cuts, IT manpower shortages, and a lack of strong centralized policy. To combat these problems NAVSUP has been aggressively pursuing a policy to consolidating their servers. During the

inventory of servers it became obvious that there was no standard for how servers or applications were maintained or hosted. Additionally, many of the program managers that were interviewed did not know enough about hosting services to be able to contract for those services. The contracts that did exist did not provide a good definition of the services to be provided, nor were there any SLAs mandating performance levels.

The SLAs in appendix (A) were developed for the program managers to assist them in the management of their post-production applications. The SLAs outline the standard hosting services that should be used across the claimancy. The intent was that these services would provide the necessary functions to properly monitor and host an application. The levels of service are broken into three levels of support: essential, enhanced and premium. The levels of service would depend upon the type of application, its criticality and fiscal constraints on the program.

The SLAs were designed to be used as a template. Each program will have to select those services and service levels that best meet the needs of their respective applications and the underlying business process. The use of a template alleviates the necessity for each program manager to research and develop hosting requirements. The template also offers services that will provide the appropriate quality and performance standard that can be used by all program managers in the claimancy. The performance thresholds were based on industry standards, or current NAVSUP standards. Program managers are expected to use benchmarking of their current services, forecasting future needs, and consulting with stakeholders to gather information to determine whether the thresholds specified in appendix (A) will meet their needs. Based on preliminary reviews most of the applications in the NAVSUP claimancy will use the standard services as outlined in the three levels of service, although some of the thresholds will have to be modified.

The SLAs are grouped in thirteen service areas that cover many of the services that were outlined in the previous chapter. Each SLA contains 17 data elements that define the service, specify the quantitative metrics that will be used to measure performance, outline roles and responsibilities, methods for collecting measurements, the threshold levels that must be met, and associated penalties or incentives.

The original intent of the dissertation was to utilize the SLAs in appendix (A) in an actual contract, and gather information from the program managers and the contractors to determine their reaction to the SLAs, their thoughts on the process of developing SLAs, and whether they felt that the SLAs were effective in delivering quality services. Unfortunately, the contract negotiations were stalled numerous times for various political, fiscal, and technical reasons. As a result, negotiations were still ongoing at the writing of this dissertation. The answers to the questions posed above would make a good follow on thesis or dissertation.

The SLAs and SOW in appendix (A) were however, used by NAVSUP in contract negotiation to compete hosting services between two organizations. Before the source-selection board met, Gartner and MetaGroup (both IT consultants) reviewed the SOW and SLAs. Both groups felt that the documents were excellent, but that the price to achieve that level of service may be too expensive. The NAVSUP source-selection board for the contract stated that the SOW and SLAs made it easy to compare the bids, as the two organizations had to address the specific services and service levels outlined in the documents. It allowed the selection board to make more of an apples-to-apples comparison. Many of the extraneous service claims from the service providers were discarded, as they did not apply to the services specified in the SOW or SLAs. Based on the estimates from the two organizations, the source-selection board applied a balanced scorecard approach and selected a winner. Unfortunately, comments and results from the source-selection board are considered proprietary, so they could not be used in this dissertation.

NAVSUP is in the process of negotiating hosting service with the winner of the source-selection board using the SOW and SLAs in appendix (A). The SLAs and SOW in appendix (A) are also being reviewed at NAVAIR, NAVFAC, SPAWAR, and NAVNETWARCOM for inclusion into a Navy-wide contract for hosting services under CLIN 0029 of the NMCI contract. To date the SLAs have received very favorable review.

H. WEAKNESSES

After evaluating the data, there were a couple weaknesses in the research used for this dissertation. The first weakness was that the questionnaire was biased toward supporting the hypothesis. The questionnaire did not go into the disadvantages of SLAs, nor did it mention case studies where SLAs were not effective. Although some negative aspects of managing SLAs were addressed in the questions in the questionnaire, all of the arguments were designed to show the user that SLAs could be used to help improve software quality and manage IT intensive systems. As the survey was targeted to IT professionals, the questionnaire was designed to present an argument that the respondents could provide comments on. The questionnaire was not intended to convince an uninformed individual of the benefits of SLAs. However, the fact that 9 percent of the respondents had less than 4 years of IT experience and over 30 percent had not dealt with SLAs before, could lead one to believe that some respondents were biased in support of the hypothesis.

In the bottom-up analysis the qualitative analysis did not consist of formal interviews with predetermined questions and documented results. Additionally information obtained from the interviews often concentrated on specific problems, and as such, the sample size contributing information on a specific topic would not be representative.

The top-down analysis also had some weaknesses. In qualitative analysis, the researcher must to some degree interact with the subject. In the top-down analysis, the researcher did not interact at all with the subjects. As such, the subjects were free to comment on the article in whatever direction they chose, and at whatever depth they determined. Although a great deal of information was obtained, this approach lacked regiment.

The survey had 43 responses. A greater number of responses would have provided more statistically meaningful data with respect to how the different groups answered the same questions. Unfortunately the size of the survey coupled with the busy schedule of most IT professionals made gathering more responses a difficult task. It

would have also been useful to add a question on whether the respondent represented public or private industry. That information may have lead to some additional insight.

Finally, the respondents were only allowed to see one example of a SLA. Respondents may have made more informed decisions if they could see a SLA for development work, maintenance, and hosting services. Unfortunately, that was not possible as the length of the survey was overly burdensome for some respondents. The amount of information presented in the questionnaire had to be weighed against the fact that fewer people would fill out the questionnaire if it became too large.

I. SUMMARY

The research utilized a pragmatic approach where both postitivist and interpretivist approaches were utilized. The survey results indicated that the respondents felt strongly that SLAs could be used to increase software quality. They also felt that SLAs helped in the management of IT intensive systems. However, comments collected from the survey indicated that SLAs, while helpful, would not be successful without upper management support.

XI. CONCLUSION

A. REASON FOR STUDY

To maintain a competitive advantage, organizations have to rely more on software-intensive information systems to support or enable their critical business processes. As a result, organizations are starting to look upon software quality management as a critical, strategic aspect of the product-development process. Despite advances in the principles and mechanics of software engineering, the quality of software is still lacking. This can be attributed in part to poor practice, including but not limited to marketing pressure, improper training, and lack of managerial oversight.

Another reason for poor quality is that contracts for outsourcing are not as explicit as they need to be. As software-intensive information systems become ever more complex and large, organizations are increasingly tempted to outsource IT development and support to companies specializing in providing IT services. While organizations are now able to take advantage of external expertise, they must write good outsourcing contracts to take the maximum advantage of that expertise. However, there are many real-world examples in which outsourcing contracts do not contain a good specification of requirements. In some cases, principle stakeholders, such as the end user, are not involved in the requirements specification activity; quality requirements are not incorporated into the requirements; and quantifiable, measurable, meaningful metrics are not identified.

Another problem leading to poor quality is that many program managers do not have the technical expertise to manage IT systems. Program managers not only need to understand the technology associated with architectures, standards, software-development processes, and software-systems engineering, but also need a firm grounding in contract management, project scheduling and tracking, risk assessment, and budgeting.

This study was conducted in an effort to determine whether SLAs could be utilized to improve software quality, and in turn, the overall quality of software-intensive information systems. This is a foundational study with the aim of determining feasibility

and collecting feedback from IT professionals on whether they believed that SLAs would improve the management and quality of IT systems. Follow on studies can evaluate the effectiveness of SLAs in providing software quality in actual projects. The SLAs in Appendix (A) have not been incorporated into the NMCI contract at this time.

B. KEY POINTS

This dissertation has explored the concept of utilizing SLAs as a tool to improve the management and quality of software-intensive systems throughout its lifecycle. We demonstrated how SLAs could be used in the requirements, development and post-production phase of software development to improve software quality. We also showed how SLAs could aid the program manager by improving configuration management, contract management, risk management, quality control, and customer satisfaction.

This dissertation demonstrated how many of the problems with software acquisition could be addressed from a software acquisition perspective. Program managers need to do more than add quality requirements to their software development contracts. In many cases requirements are not measured until the end of a major milestone, and if there are any problems with the requirements, the program managers have little recourse short of canceling the program. Although SLAs are also requirements, their format makes them a more effective contracting tool. SLAs provide a detail description of the services, service levels, and the method to measure and monitor the service level. SLAs are also more effective because the measurement period is short enough to resolve problems and the penalties in the SLAs give the program manager recourse if quality levels are not met.

SLAs can help to improve quality in the various phases of the software lifecycle. In the requirements engineering phase of software development SLAs help to bring all stakeholders together to focus on identifying quantifiable quality factors that they feel are essential in a system to support the underlying business process. SLAs specify the quality metrics and quality thresholds that allow an organization to determine whether quality requirement have been met. As such, SLAs make explicit many of the quality factors that users may implicitly assume. Measurements and monitoring resulting from

SLAs also support early detection and resolution of quality problems. SLAs help reinforce the notion that quality management is a strategic, critical aspect of the quality control process throughout a system's lifecycle.

In the development phase, the quality factors that are addressed in the SLAs drive architectural and design decisions. If developers know which of the characteristics are most critical to project success they can select the architecture, design, and programming approaches that best achieve the specified quality goals. SLAs help ensure that quality is designed in at the beginning phases of the lifecycle. SLAs can also improve software quality in the development phase by contractually mandating that certain quality control measures (e.g., adhering to specified standards and processes) be performed.

In the post-production phase of software development SLAs can be used to specify the quality requirements for application performance, software maintenance efforts and hosting services throughout its lifecycle. Monitoring the performance of the application and its supporting infrastructure once it is deployed is essential in implementing process and quality control, as well as maintaining customer satisfaction.

It requires a great deal of management to produce quality software. Program managers have to ensure that quality considerations are addressed early in the lifecycle and they must provide the proper amount of oversight to ensure those quality factors are incorporated into the final product. SLAs provide quality control measures that can assist program managers in many of the managerial tasks necessary to ensure quality is delivered in the final product.

Program managers need to measure and monitor contractor, project, and system performance throughout the project's lifecycle to ensure requirements, standards, processes, and quality requirements are being met. SLAs mandate monitoring of the quality requirements associated with process, product, and project quality. If quality levels are not met, program managers and the contractor are informed of the violation and potential risks, allowing them to take the action necessary to correct the situation.

The thirteen SLAs in appendix (A) illustrate how SLAs could be used in the post-production phase of software lifecycle to assist the program manager by establishing process and quality control measures necessary to support a software-intensive system.

The SLAs in appendix (A) introduced a new format that was useful in coupling the quality requirements back to the business processes they supported. If used properly, the new SLA format improved on standard SLA formats by provided greater detail with respect to the services required, the means of measuring the services and the responsibilities of all parties.

The survey of IT professionals indicates agreement that SLAs can play an important role in addressing software quality. SLAs can drive product, process, project, and deployment quality solutions. SLAs can help ensure that quality requirements are established early in the development cycle in order to be incorporated into preliminary designs. SLAs help program managers with the oversight of the various aspects of the projects. SLAs also carry sufficient weight through penalties and incentives to focus management and contractor attention on the quality issues that will impact business critical areas.

C. FUTURE WORK

Although SLAs are not uncommon in application-hosting services, they are not usually found in software-development contracts. There are a number of areas that can build upon the work conducted in this dissertation.

1. Evaluation in Actual Contracting

Future study is necessary to determine the magnitude and direction of effects of utilizing SLAs in actual contracts for host services, as well as application development. Studies can evaluate how well the SLAs helped in requirements engineering, design, post-production support and program-management tasks. This research can also evaluate whether SLAs helped in the negotiation and source-selection process, or whether they complicated the contracting process. These studies should also focus on the reactions of program managers, end users, and contractors, as well evaluating upper level management's support of the SLAs, and whether they believe SLAs are effective tools for quality control.

SLAs in theory should lead to higher levels of software quality, but additional research utilizing actual contracts is needed to test the hypothesis proposed in this

dissertation. If quality is defined as the extent to which a system, process or component meets specified requirements and meets user needs, studies can be conducted to compare similar software projects with SLAs against those without SLAs to determine if the SLAs improved quality. Future studies can also evaluate the success or adoption of the concept of SLAs within public and private organizations. Studies can also focus on the effectiveness of SLAs on large and small software-development projects.

Template SLAs, such as those found in Appendix (A), are designed to assist program managers that may not have the technical skills necessary to lead the SLA development effort on their own. Program managers can modify the existing template SLAs to suit their application requirements. Additional research can focus on the effectiveness of template SLAs. Studies can evaluate whether template SLAs helped the program manager, contractor, or end users incorporate quality requirements into the software specifications.

2. Quality Factors

Although there has been a great deal of research on software metrics, few models have been widely adopted in the commercial sector. There are no industry-accepted standards that define quality factors, quality metrics, and their associated quality thresholds. In many cases quality models are not used because automated tools do not exist to make measurements easy to gather, or because the measurements are too subjective to be of value outside of a particular organization.

Research is needed to determine the quality factors, their associated quality metrics, and meaningful quality thresholds that can best measure product, process, and project and post-production quality. Studies are needed to determine which quality models and quality attributes are best suited for different types of IT systems (e.g., missile systems should have high reliability and response rates, whereas logistics systems should have high interoperability, portability, reliability and usability). These studies should concentrate on quality models and metrics that can support commercial software development. A measurement of cyclic complexity of X means little to commercial developers unless there is a cause and effect associated with a measurement of X. For example, organizations with project complexity between X and Y have a sixty-five

percent failure rate (cost and schedule overrun) as demonstrated in over 500 software projects analyzed for cyclic complexity.

Follow-on research can also concentrate on writing template SLAs for product, process and project quality. This research can evaluate various quality models and determine which can be utilized in SLAs to encourage the adoption of processes leading to a quality product. The research would not only identify potential quality factors, but it would also have to identify quality metrics and thresholds that could be utilized in SLAs. Research can also improve the template SLAs in Appendix (A) that were written for hosting services.

3. Availability

In Appendix (A), we discussed availability in the context of ability to compute. Current monitoring tools such as Tivoli and HP Open View can provide a wealth of information concerning server and network performance, but it is difficult to determine which metrics warrant the most attention, and what quality thresholds are acceptable. There is an ongoing debate among system and network administrators as to which metrics are most important. For example, if CPU utilization in a server is important, should the system administrator take action when the utilization is eighty percent, ninety percent, ninety-five percent, or higher? Appendix (A) lists some common quality metrics and thresholds, but further research is needed to determine an industry-accepted list of quality factors that represent an ability to compute.

4. End-to-End SLAs

SLAs are most meaningful when the measurements come from the end user's perspective; however, end-to-end SLAs are difficult to achieve. More research is needed to generate tools or processes that will easily allow end-to-end measurements for response time, availability, and other quality metrics across an infrastructure that is owned by different entities.

There are currently tools that can account for end-to-end response times, but agents are needed within the client and server side of the application to properly account for where delays occur. Research can concentrate on other approaches such as coding the application to send timestamps for certain test inquires.

APPENDIX A: NAVSUP HOSTING REQUIREMENTS AND SERVICE LEVEL AGREEMENTS

Abstract

This paper consists of a statement of work (SOW) and its related service level agreements (SLAs) for hosting services. The paper will be used as part of contract negotiations to outsource the hosting functions for NAVSUP owned applications. The SOW contains the hosting requirements that NAVSUP believes are necessary to support the application.

NAVSUP will maintain control and responsibility of the application software, but all server and infrastructure hardware as well as system software support (operating system, monitoring software, utilities, and infrastructure software), is the responsibility of the service provider. The SOW details hosting requirements at three levels to allow program managers to select the levels and the corresponding services that best meet their needs.

A service level agreement (SLA) is an agreement between a provider of services and a customer that defines a level of performance. This agreement defines in measurable terms the service to be performed, the level of service that is acceptable, and the means to determine if the service is being provided at the agreed upon levels. SLAs define the quality of service, and how it is measured. There are fourteen SLAs defined that support the SOW.

This paper provides a starting point for negotiating host services. The intent of this paper is to give the program managers a document that listed hosting services that will provide a high level of support for their application. The SOW and SLA were designed to meet the needs of most applications, but each program manager will have the flexibility to select and modify the services and service levels required to support their specific applications.

NAVSUP Hosting Statement of Work

The scope of this document is to define the requirements for hosting Navy midrange application systems. Midrange systems are defined as those systems that fall between stand-alone applications residing on a personal computer (PC), and those that reside on a mainframe computer. The scope assumes the Supplier maintains ownership of the servers, networking hardware, and associated systems software that is necessary to provide the hosting environment. It is not the responsibility of the Supplier to purchase or maintain application software unless otherwise negotiated between the Navy's Application Program Manager and the Supplier. The scope does not include hosting hardware that is owned by the Navy, which is referred to as co-location services. Although many of the requirements in this document apply to co-located hardware, co-location services are not part of this document and will be negotiated separately between the Navy and the Supplier. The government is contracting for a hosting service. The government does not intend to procure or maintain any of the hardware in the host

environment. The Supplier is responsible for the hardware hosting the application. That allows the Supplier the flexibility to maximize efficiencies within their organization, resulting in a lower cost to the government.

This document is intended for production applications. It does not apply to test platforms, although this document can be easily modified to support that need. Test platforms will be negotiated under another contract vehicle with appropriate service level agreements (SLAs).

This document attempts to draw a clear line between application support, which is the responsibility of the program manager, and system software support (operating system, monitoring software, utilities, and infrastructure software), which is the responsibility of the Supplier. Any application support, other than monitoring, is outside the scope of this contract.

A. ESSENTIAL PACKAGE SYSTEM SUPPORT AREAS

This statement of work (SOW) outlines three levels of support, the essential package, enhanced package and the premier package. The application's support requirements will dictate which package should be selected. If the enhanced package is selected, all of the services included in the essential package will also be included in the enhanced package. The premier service will also include services outlined in the enhanced package.

In addition to the services offered by each package, specified services can be added or deleted from the package. Adjusted services are outlined at the end of each package description.

The essential package is designed for stable, non-critical applications with minimal requirements for change, and predictable growth. As such, the services will reflect predictable capacity utilization, a consistent user base, and reliable application software.

1. Application Migration Service

Application Migration Services are the tasks necessary to transfer an application from one host environment to another. This seemingly simple task can be extremely complicated and difficult. A well-defined process needs to be implemented to ensure a successful migration. Migration services include information collection, platform and environment design, execution planning, testing, and ultimate deployment of the application.

a. Midrange Site Transition Services

Midrange site transition services must be available for moving Navy applications into the host environment. These services must include the use of a proven project management methodology and proven experience with transitioning similar applications.

Midrange Site Transition Services Requirements are:

- The Supplier will gather information on the application, develop a design plan for hosting the application, perform testing in accordance with the test plan, redesign if needed, prepare for ongoing production support services and deploy the application in a production environment.
- The Supplier must obtain, assemble, install, customize deploy, and tune network and server hardware, operating systems, and associated applications.
- The Supplier must coordinate with Navy Program Managers and technical staff to perform requirements determination and obtain a site survey of the application system being transitioned.
- The Supplier must develop a risk assessment plan. The Supplier must work with Navy Program Managers to identify and mitigate the risks associated with the transition of the application into the hosted environment.
- The Supplier must provide a project manager to oversee transition execution.
- The Supplier must provide a project plan with extensive detail, a work breakdown structure, and timelines to enable the execution to be managed and executed effectively within the Navy's operational constraints and business requirements.
- The Supplier must test the project plan execution in a test environment to validate the documented process and to confirm the defined production infrastructure supports the application and integrates into the host environment.
- The Supplier will work with the Navy application development team in developing a test plan to ensure the application performs as expected in the host environment. The Navy must approve the Supplier's test plan. The plan must outline the various tests to be performed, and establish thresholds for success. The Navy Program Manager must be responsible for functional testing, or for developing test scripts.
- The Supplier must ensure that the application's performance in the new production environment is equal to or greater than the performance the application demonstrated before the transition. Benchmark tests will be performed in both environments for comparison.
- The Supplier must test the application in a test environment before moving the application into production. The test of the application must follow the processes defined in the test plan. The test plan must ensure the testing environment emulates the application's production environment.
- The Supplier must provide project status or updates (at least weekly) of the plan from development through to implementation and post-migration.
- The Supplier must be able to execute the transition using a proven and repeatable set of processes that include multiple implementation options based on Navy requirements.
- The Supplier must provide a design solution for the hosted applications and be able to implement the solution.
- The Supplier must review implementation requests and the platform solution design with Navy Program Managers to verify the requirements, educate developers or maintainers on the technology being employed, and ensure they understand the new architecture.

- The Supplier must interact with the identified network provider to help confirm that the platform configuration integrates the network requirements and connectivity is established to the Navy's LAN/BAN/WAN.
- The Supplier must ensure that applications that print to network printers have the necessary connectivity to the network and that the printer is properly set up on the server.
- The Supplier must verify that the appropriate hardware and system-level software products, for example, the operating system and non-application software, are obtained and ready to implement before the transition begins.
- The Supplier must work with the Navy technical staff to obtain, install and configure the application being transitioned.
- The Supplier must communicate migration support issues or implementation concerns through the site-specific communication process. The Supplier must provide progress reports to the Navy Program Manager as required.
- The Supplier must install and configure system-level software according to requirements defined in the platform solution design.
- The Supplier must work with the Navy Program Manager to define the backup and recovery needs for the application being transitioned.
- The Supplier must obtain signoff from the Navy Program Manager before going live with the application in the new environment.
- The Supplier must provide a final review of the implementation to determine whether the requirements have been met. Based on the final review, a production implementation live date is agreed to, at which point Transition Services end.
- The Supplier must incorporate the new application and associated hardware and software into all necessary documentation (e.g., hardware and software configuration documents, the backup plan, the disaster recovery plan, operation procedures, network diagram, etc…)
- The Supplier must complete a vulnerability assessment of the host environment (hardware, software and supporting infrastructure) that will be used to host the application. The information will be incorporated into the Supplier's System Security Authorization Agreement (SSAA) in accordance with the DoD Information Technology Security Certification and Accreditation Process (DITSCAP) program outlined in DoD Instruction 5200.40 to cover the host environment. This requirement is also included under the security section in more detail.
- The Supplier must provide the following documentation to the Navy Program Manager upon request: Project Plan, Risk Assessment Plan, Initial Configuration Audit, Design Solution, Results from initial audit of the application and the requirements determination, Backup Plan, Disaster Recovery Plan, the Test Plan, and SSAA documentation.

2. Systems Management

Systems Management is the process of monitoring, evaluating, and reviewing the compute operation to determine whether operational requirements are met. The system

management services included in the Essential Package are host system and network monitoring, performance monitoring, intrusion detection, automating compute operations, and system backup and recovery.

a. System and Network Monitoring

The System and Network Monitoring Services provide the operational support processes and procedures required for monitoring midrange compute environments for delivery of a stable, reliable functional environment.

System and Network Monitoring Services Requirements are:

- Monitoring of all network hardware (including firewall) must comply with NMCI, DoN, and DoD guidance and regulations.
- The Supplier should monitor application software status to determine if the application is responding.
- The Supplier must monitor all systems hardware and systems software that are used to support the application systems being hosted. Exclusions are listed below, however, monitoring for services on the list of excluded services must be available as a separate offering where indicated.

  Exclusions are:

  - The Supplier should monitor application databases for space utilization and database performance and other specific database criteria such as dead locks (available under enhanced services).
  - The Supplier should monitor applications database to ensure the database is responding to requests (available under enhanced services).
- The Supplier must monitor all system consoles and logs. Console monitoring must be done using industry standard procedures and industry standard software. Some examples of industry standard monitoring software are: HP Openview, Cisco Works, CA-TNG, and NetScout.

  Console Monitoring Includes:

  - The Supplier must implement Event Detection Monitoring on the servers to detect any message sent to the system log and then cause an automated event to occur.
  - The Supplier must implement Network Monitoring on network assets within the host environment. Some of the monitoring functions include quality of service analysis, pinging an IP address or collecting data from an SNMP device on the network resulting in an automated event.
  - The Supplier must implement automated notification for console event alerts (e.g., e-mail, alarms, automatic trouble ticket generation).
- The Supplier must monitor network bandwidth for each application.
- The Supplier must monitor network bandwidth for the host environment network.
- The Supplier must monitor IP availability for each machine. Furthermore, selected sites on the Internet must be periodically (hourly) pinged to alert the staff to potential Internet problems.
- The Supplier must monitor web sites for hosted applications.

Web Site Monitoring includes:

- The Supplier must monitor polling of the Web site index (main) page.
- The Supplier must implement automated notification for console event alerts if the site does not respond.
- The Supplier must provide reports using a standard reporting tool on web site activities of the hosted applications (popularity documents, SLA compliance, report of the sites that access the user's Web server most often, etc).
- The Supplier must provide monthly URL availability reports, if applicable, for the hosted application.
- The Supplier must monitor URL availability to check the correct function of HTTP processes at timed intervals as specified by the Navy Program Manager.
- The Supplier must monitor HTTP response times. A threshold will be set on a site-by-site basis; the party responsible for support is notified if the threshold is exceeded.
- The Supplier must monitor HTTP Process Availability to ensure processes operating on the Web server do not have "out-of-bounds" conditions that may indicate an immediate or potential problem.

b. Performance Management

Performance Management processes include defining reasonable and measurable performance metrics, documenting and executing performance monitoring methods, maintaining contingency plans with corrective actions for exception performance, maintaining a support plan that incorporates the appropriate performance monitoring of documented requirements, reporting, implementing the monitoring activities, and measuring ongoing results.

Performance Management Services include the support processes to collect, monitor, and analyze system performance information, including, but not limited to:

- Processor(s) usage
- Input/output (I/O) throughput activity (e.g., operating system response time, disk access times, transfer times to disk, backplane speed, paging)
- Disk usage
- Memory usage

As needed, performance changes are implemented according to a change management process to modify the configuration and tune the system to optimize the effectiveness and efficiency of the midrange environment.

Performance Management Requirements are:

- The Supplier must maintain operating system parameters to manage performance and workload throughput. This includes tuning the system in the attempt to optimize the application's performance.
- The Supplier must monitor CPU, memory, I/O, and disk utilization against predetermined thresholds.

- The Supplier must monitor predetermined exception thresholds for Network bandwidth to assist in establishing monitoring alerts.
- The Supplier must provide monthly reports on CPU, Disk, and Memory utilization.
- The Supplier must provide monthly reports on network bandwidth and utilization.
- The Supplier must manage predefined exception thresholds for the operating system and major components to assist in establishing monitoring alerts.
- The Supplier must monitor real-time performance using system management tools to resolve system resource and performance problems.
- The Supplier must collect performance data dynamically to assist in problem determination.
- The Supplier must analyze historical performance data to isolate or identify potential performance issues.
- The Supplier must be able to recommend and implement workload allocation changes as they relate to applications use of server and network resources to assist the Navy Program Managers in resolving performance problems.
- Historical performance data will be retained for 1 year for trend analysis.

c. Capacity Management

Capacity Management Services include planning and monitoring system usage and capacity, both short-term and long-term, forecasting resource requirements, and analyzing and reporting resource trends. The Supplier's capacity processes should use metrics and reports that enable a clear understanding of overall performance and trends.

The Capacity Management Services Requirements are:

- The Supplier must perform resource usage analysis, including tracking, trending, and graphically illustrating resource usage by CPU, memory, I/O, storage, and tape consumption.
- The Supplier must provide reports, at least monthly, to the Navy Program Manager that show standard resource usage, trending and analysis. The Supplier must assist the Navy Program Manager in understanding the hosted applications current resource usage and future resource needs.
- The Supplier must use capacity planning to project the effects of new business and workload changes as needed. For example, the Supplier will perform capacity modeling when new business or application growth is anticipated, when substantial changes to existing business are anticipated, or when substantial configuration (hardware/software) changes are performed within the systems.
- The Supplier must take appropriate action to mitigate resource problems, including increasing the necessary resources. Additional resources needed to directly support the application as a result of an application change must be addressed at the Change Review Board. The Supplier will provide cost information associated with

resource changes resulting from an approved application change. If the application change is approved, the program will be charged for the additional resources identified.

d. System Operations Automation

System Operations Automation Services include the use of Industry Standard automation software that provides for the automatic monitoring and remote reconfiguration of system environment resources or files to achieve operational efficiencies. Examples of Industry Standard automation tools are CA-TNG and HP OpenView.

System Operations Automation Services Requirements are:

- The Supplier must perform problem determination, day-to-day maintenance, and support for automation products and operational processes.
- The Supplier must be able to customize the automation requirements based on contracted services.
- The Supplier will continuously identify opportunities to remove manual interventions for ongoing support services.
- The Supplier will review automation software to ensure that they reflect the most recent policies and procedures.

3. Software Management

Software Configuration Management Services provide and maintain software for the operating environment, including operating system software and related system software. As part of these services the Supplier must perform the basic operating system software tuning that is required to maintain day-to-day operations.

a. Configuration Management

Configuration management involves the steps necessary to review and document changes to both the system software and the application, so that program manager and the Supplier are aware of maintenance or upgrades that may affect their application, or support processes. Accurate software configuration is essential when troubleshooting errors, performing software maintenance, and developing software (test beds should emulate production environment). Changes to the hardware or system software that impact the operations of the network, the servers, or the application must be reported to the Change Review Board (e.g. router configuration changes to close specific ports, or adding monitoring tools that impact server resources.)

The Change Review Board is chaired by the program manager for the application. The Change Review Board consists of the program manager, design personnel, functional experts (if necessary), a representative from the Supplier's organization, government Information System Security Manager (ISSM) to address information assurance issues, and other personnel deemed necessary by the program manager or their chain of

command. The intent of the Change Review Board is to approve any hardware or software configuration changes. The program manager and designers need to know if the Supplier's proposed changes will impact the application, or architecture. The Supplier must know if proposed application changes will affect resources, monitoring software, and network bandwidth. Additionally all approved changes are documented, improving communication channels, and ensuring only approved changes are implemented.

Configuration data will be held in a central repository that is web accessible. The repository will be populated using industry standard COTS packages, such as PVCS. The same configuration software should be used for all Navy applications.

Software Configuration Service requirements are:

- The Supplier must maintain documentation of server and network software configurations including OS release levels, configurations, patches, etc.
- The Supplier must, in coordination with Navy Program Managers, maintain documentation of application configurations including application software release levels, configurations, patches, etc.
- The Supplier must maintain documentation of all changes approved by the Change Review Board including date approved, change summary and date change applied.
- The Supplier must make all documentation available to the Navy upon request. The Navy program manager's staff will have web access to view configuration data held in the central repository.

b. System Product Integration and Problem Resolution

The Supplier must integrate the software components of the operating system and various third-party software products. System Product Integration and Problem Resolution provide the operational processes necessary to maintain a stable operation environment to meet the Navy's application specific operational requirements.

System Product Integration and Problem Resolution Services Requirements are:

- The Supplier must perform the planning, installation, testing, and upgrading of system-level software, such as operating system and other non-application software, or application software requiring super user access.
- The Supplier must perform problem resolution including problem determination, interface, and escalation with third-party suppliers, if necessary, to correct system component problems.
- The Supplier must participate in identifying system product problems including connectivity and associated network problems.

c. System Software Maintenance

System Software Maintenance Services provide ongoing maintenance and support for the software supporting the application. These services also provide

preventive software maintenance services when required. System software also includes maintenance to the infrastructure (e.g., routers, firewalls).

System Software Maintenance Services Requirements are:

- The Supplier must assist the Navy technical support staff with installing applications software when root/Administrator access is needed and when loading the application software media into the hosted server.
- The Supplier must review product status and maintenance information for system patches to identify current version information and potential problems. All patches should be installed, unless there are mitigating circumstances. The program's Change Review Board must be notified of patches to be installed, and those patches that will not be installed.
- The Supplier must install preventive maintenance (e.g. software updates, software releases, and virus and anti-spam updates) to supported system software products to prevent known problems from impacting the operating environment.
- The Supplier must implement a permanent corrective action with appropriate monitoring procedures to ensure software faults are eliminated from the operating environment.
- The Supplier must communicate changes that require system down time to the Change Review Board. In the case of emergent changes that effect system availability the Supplier must notify the Navy Program Manager. If the change cannot wait for approval, the Supplier should notify the Navy Program Manager and the Change Review Board as soon as possible.
- The Supplier must ensure that the application has proper licenses for COTS products that are incorporated into the application. This includes accounting for usage-charged types of software agreements.
- The Supplier must review the Navy Program Manager's software service and licensing agreements and provide recommendations. Application consolidation may allow program manager's to reduce or eliminate some third party software requirements.

d. Software Refresh

Software refresh (system software, not application software) ensures that the software supporting the application does not become obsolescent. Technology is evolving at a rapid pace, and software must be updated to take advantage of new technology.

Software Refresh Services Requirements are:

- The Supplier must plan for, install, and support new operating system, infrastructure and related system software. The plan must include the steps necessary for a successful migration of the application systems software.
- The Supplier must maintain a test system for systems software.
- The Supplier must work with the Navy Program Managers and Navy Technical Staff to research and resolve software compatibility issues allowing migration from the current suite of products to upgraded products and releases.

- The Supplier must have a documented software refresh plan. Some legacy applications currently in production have dependencies that do not allow for systems software upgrades and therefore should be exempt from this requirement. The application systems that should be exempt and their dependencies will be provided by the Navy Program Manager on an application-by-application basis. The refresh plan will have to be agreed upon with the Navy Program Manager and will have to take NMCI desktop systems into consideration.
- The Supplier must work with the Navy Program Managers to identify software changes that may impact applications. The Supplier will then work with the Program Manager to create a test plan, if necessary, to confirm that changes in software functionality do not adversely impact an application. The Supplier must address these changes with the Navy Program Managers at a meeting of the Change Review Board.
- The Supplier must design the necessary back-off processes to restore to the former operating environment if unforeseen problems occur.

4. Hardware Management

Hardware Configuration Management provides services for installing and maintaining the compute configurations to meet changing requirements for compute resources and maintains the configuration plan to meet application specific requirements.

a. Hardware Configuration Management

Configuration management involves the steps necessary to review and document changes to hardware used to support the application, so that program manager's staff is aware of changes that may affect their application.

- The Supplier must present hardware changes to the Change Review Board (CRB). Hardware changes resulting from hardware vendor requirements will still have to be briefed to the CRB.
- The Supplier must maintain documentation of hardware configurations, including equipment placement, network diagrams, cabling, connectivity details, application mapping, disk partition information, peripherals, etc.
- The Supplier must address new hardware installations or modification at a meeting of the Change Review Board.

b. Hardware Support and Maintenance

Hardware Support and Maintenance Services provide the support services necessary to ensure compute equipment is maintained, and operational.

Hardware Support and Maintenance Requirements are:

- The Supplier must monitor midrange compute hardware, including processors, storage, and peripherals for malfunction.

- The Supplier must coordinate trouble-shooting, repair and, if necessary, escalation of hardware-related malfunctions with the hardware support vendor.
- The Supplier must manage hardware maintenance requirements based on the manufacturer's recommended schedule.
- The Supplier must coordinate and provide installation support hardware corrective maintenance requirements with hardware vendors.
- The Supplier must maintain documentation of all hardware changes approved by the Change Review Board including date approved, change summary and date change applied.
- The Supplier must make all hardware configuration documentation available to the Navy upon request.
- The Supplier must include a schedule for maintenance downtime. The downtime will abide by timeframes and duration specified in the service level agreements.
- The Supplier will have a documented preventative maintenance program for hardware support.

c. Hardware Refresh Services

The Supplier is responsible for replacing existing hardware components to include firewall, network, servers, etc. The Supplier will determine the hardware refresh rate, based upon their ability to meet requirements outlined in the service level agreements.

Hardware Refresh Services Requirements are:

- The Supplier must have a documented hardware refresh policy that includes migration strategies, timelines, accessibility, etc.
- The Supplier must coordinate planning, installation and testing, including shipping and receiving, of midrange compute hardware and environmental equipment.
- The Supplier must create a complete migration project plan and timeline and present the plan to the Change Review Board for approval.
- The Supplier must coordinate testing activities for the hosted applications with the effected Navy Program Managers.
- The Supplier must manage data migration and data movement processes, where possible, based on current hardware and software configuration to enable storage asset replacement.
- The Supplier must update documentation of hardware configurations, including equipment placement, cabling, and connectivity details as hardware configurations are refreshed.

5. Security Management

The Supplier must provide Security Management Services to protect the confidentiality, integrity, and availability of the Navy's information assets. The Services must adhere to all DoD, DoN policies and procedures (appendix (c) provides a list of

relevant information assurance policies). Services include supporting data integrity protection software, user identification maintenance (authentication services), and password issuance. Server security must be monitored 24x7x365 unless an adjustment is made to the business hours of operational support coverage. Network security must be monitored 24x7x365 regardless of any adjustments. Physical security requirements for the hosting facility are defined as part of the Facilities requirements in the Enterprise Foundation section.

Security Management Services only address those areas that deal directly with the network, servers and associated hardware that support the Navy's application systems and do not address access to the application systems themselves. For instance, the Supplier must provide an identification and authentication mechanism for access to the application, but will not address or control identification and authentication mechanisms that allow access into the application itself.

The scope of these services includes the entire server farm from the firewall to the actual server. The firewall protecting the server farm is inside the scope of this SOW. The network from the end-user to the host environment firewall is not within scope for this SOW.

a. Security Management Services

- The Supplier must implement the appropriate INFOCON conditions when dictated by designated Navy personnel. The end users within NMCI must be able to maintain connectivity with the application during all INFOCON conditions.
- The Supplier must ensure that all personnel with access to government information have received the proper clearance from the government. Personnel without proper clearance will not be authorized access to any government data, nor will they be allowed to monitor any government applications.
- The Supplier must implement Root/Administrator Access Restriction/Verification – Access is restricted to a known set of Supplier support personnel.
- The Supplier must provide Vulnerability Scanning that identifies vulnerable configurations settings on network/system components, as well as identifying unauthorized ports/protocols and their associated applications. The scans must be periodically reviewed to provide a secure environment.
- The Supplier must run periodic (once a shift) scans against systems comparing current file permissions against an approved baseline.
- Security logs (server, firewall and network) will be reviewed once a shift at random hours. Although log entries can be sent to a central monitor it is necessary to physically review logs to discern patterns that may not be automatically detected.
- The Supplier must ensure that access to system-level files and services be restricted by use of operating system-level file permissions. The Supplier must maintain a database listing users, their access and permissions, their roles and security level.

- The Supplier must ensure that access through routers and firewalls adhere to the NMCI, DoD, and DoN Network requirements as they relate to protocols and specific IP address or ranges.
- The Supplier must ensure that security changes are processed, reviewed, tested and approved by Supplier and Navy Change Review Board before implementation.
- The Supplier will use base DoD and DoN configurations for server and network installations when they are available.
- The Supplier will configure each system platform based on a government supplied secure configuration guide. IAVA/B/TA will be implemented as required by DoN. Attachment (b) provides the listing of Secure Configuration Guides.
- DoD System Administrators will be properly trained and certified in accordance with the Office of the Secretary of Defense (DoD Memorandum dated 29 June 1998). This is a requirement for government agencies only.
- The Supplier is responsible for revoking all access rights and privileges of the Supplier's employees that were transferred, are retiring, or have been terminated. The Supplier must notify the Navy Program Manager that those individuals are no longer working on the project.
- The Supplier will provide a security point of contact or contacts to interface with the government on matters relating to information assurance issues.
- The Supplier will provide government access (customer, Naval audit) to the applicable information assurance documentation (logs, procedures) in accordance with the Government Information Security Reform Act (GISRA) with is part of section 811 of the Defense Authorization Act.
- The applicable System Administrator for each platform/system will maintain a repository of access request forms and user agreement forms for administrator accounts for their platform/system. The application administrator will maintain a repository of access request forms and user agreement forms for user accounts.
- The applicable platform/system systems administrator will ensure all non-public web sites implement identification and authentication mechanisms (e.g., user id/password, DoD PKI certificate, CAC card with hardware certificate), and are SSL enabled with a DoD PKI server certificate. The systems administrator will ensure the server certificate is renewed prior to expiration date.

b. Intrusion Detection Services

The Supplier must incorporate Intrusion Detection Services using an Intrusion Detection System (IDS) that is designed to monitor the network for known security threats.

Intrusion Detection Services Requirements are:

- The Supplier must implement an industry standard (NSA approved) IDS that enables real-time notification of potential security problems, such as denial-of-service attacks or other security breaches.
- The Supplier must implement an industry standard (NSA approved) IDS that monitors inbound network traffic for numerous attack signatures. In the event of an

intrusion alert, the Supplier must be automatically notified and appropriate action must be taken based on the alert's nature.

- The Supplier must implement the most current versions of software that recognize activity patterns of known attack signatures.
- The Supplier must provide monthly reports of security incidents to the government.
- The Supplier must notify the affected government Program Managers if an intrusion is successful and provide an assessment of the damage.
- The Supplier must have sensors in place that monitor network traffic and search for known attack signatures.
- The Supplier must use agents that monitor the network and analyze audit logs and search for attack signatures and policy violations.
- The Supplier must have a console to remotely manage the sensors through authenticated and encrypted communications.
- The Supplier must use an automated incident response capability that may reconfigure firewall rule sets to repel an attack.
- The Supplier must use automated notification to administrators in the event of an attack.
- The Supplier must utilize authenticated and encrypted (128-bit) communications between sensors/agents and consoles.
- The Supplier must ensure that sensors/agents are hardened from attack. This is usually done by ensuring the integrity of the software through products that create an encrypted hash of the file.
- The Supplier must notify the affected government ISSM and program manager within 30 minutes if an incident causes service degradation/disruption or if a successful intrusion occurs. The Supplier will complete the Navy Incident Report (see appendix c) with assessment of the damage, and provide a copy to the ISSM in accordance with the timelines outlined in instruction OPNAVINST 2201.2.
- If an intrusion is successful, the Supplier will notify the appropriate government personnel and activities within the timeframes established in the SLA.

c. Vulnerability Assessment

The Supplier will have a developed perimeter vulnerability assessment methodology specifically designed to determine an organization's overall vulnerability to Internet-based attacks, along with identifying exposures and risks associated with any of the organization's firewalls, FTP servers, Web servers, DNS servers, and e-mail servers residing on their Internet perimeter.

This assessment will run remotely, probing the Internet/Intranet perimeter for all hosted applications in the same way a "hacker" would. The process will identify weaknesses in the hosted network and system configurations, thus providing the capability to immediately address and correct any identified deficiencies or shortcomings.

This vulnerability assessment is separate from the "red team" assessment, which is a government-funded assessment. Service Level Agreements will dictate the metrics

used to determine compliance with regard to the "red team" assessment. The assessments discussed in this section will be undertaken by the Supplier to prepare for the government assessments.

Vulnerability scanning will assess system vulnerabilities from two perspectives: network vulnerabilities and host-based vulnerabilities. Network vulnerabilities are those weaknesses in systems and network components that could be exploited by an attack originating outside the system, including IP spoofing, TCP/UDP port attacks, SYN floods, and other denial-of-service attacks. Host (operating system) vulnerabilities are weaknesses in systems that could be exploited at the system itself, including poor authentication, easily guessed passwords, and poor access control lists. System vulnerability detection also investigates system vulnerabilities on primary service entities such as servers, routers, and firewalls.

- The results of all Vulnerability Assessments are classified in accordance with the appropriate classification guide. The Supplier must provide personnel with the appropriate security clearance to conduct and review the assessments and produce a corrective action plan based on the results of the Assessments.
- Port Scanning runs an in-depth port scan of the platform on the host environment's Internet perimeter to identify "high-risk" services found running on the hosts visible to the Internet. The Supplier will take action to mitigate the risks associated with those ports.
- Vulnerability Assessment Scanning uses a variety of automated and commercially available tools to remotely probe the specified networks for security vulnerabilities, known software bugs, configuration problems, and unnecessary services, uncovering security weaknesses.
- The Supplier should also provide a periodic review of systems and administrative security controls to make sure that they meet or exceed NMCI, DoD, and DoN standards. The review is required to make sure that all changes made to security control mechanisms can be traced to a duly authorized security change request.
- Server Vulnerability Assessment is a service designed to determine vulnerabilities, exposures, and risks associated with the Navy's specific server(s). This will include completing a System Security Authorization Agreement (SSAA) in accordance with the DoD Information Technology Security Certification and Accreditation Process (DITSCAP) program outlined in DoD Instruction 5200.40 to cover the host environment. The SSAA will be made available to the Navy Program Manager for incorporation into their systems' SSAA. The application specific information required by the SSAA is the program manager's responsibility. The application specific information will be shared with the Supplier to ensure that the Supplier is aware of possible security problems that may affect the host network, systems, or other applications. If an application evaluation is necessary to complete the application's SSAA, that task will be negotiated separately.
- The Supplier must ensure that vulnerability scanning adheres to all DoD and DoN security policies and procedures as they pertain to Networks and Servers.
- The Supplier must run the vulnerability assessment directly on the Web and application server(s), scanning the configuration for known security weaknesses.

- Supplier personnel must review the results of the server scan and provide a summary of findings to the Navy.
- The Supplier must provide one annual vulnerability scan run on the Navy's server – occurs just before the site goes online (LIVE URL); all other scans must occur at a minimum of annually.
- Real time Terminal in-state Residency (TSR) antivirus software protection will be implemented on each system to protect against malicious code as a result of file uploads/downloads.
- For DoD owned co-located servers, the DoD antivirus protection software may be used (DoD has already paid for an enterprise license).
- The Supplier must implement and maintain industry standard anti-spam software on servers running SMTP or E-Mail gateways.
- Upon report of an incident affecting the government application, the Supplier will allow FIWC to perform an Online survey (OLS) on the applicable network where the incident occurred. The OLS is an external probe that attempts to recreate the incident, or test to ensure the vulnerability that was exploited is corrected.

d. Data Protection Software Services

The Supplier will use Data Protection Software Service to ensure the integrity of essential data files. Data integrity processes and procedures will be in accordance with DoD, DoN policies.

Data Protection Software Service Requirements are:

- The Supplier must install, maintain, and administer security system software that controls user access to information on a midrange server platform, such as access control lists.
- Files containing passwords must be protected at the same level of protection as the most sensitive asset it protects or as "sensitive but unclassified data", whichever security level is higher.
- The Supplier must have processes, procedures and tools to maintain essential operating system and related system software data integrity.

e. User Identification (ID) Maintenance and Password Issuance

These services ensure only authorized users have access to their requested files and unauthorized access is denied without hindering business practices.

User ID Maintenance and Password Issuance Services Requirements are:

- The Supplier must use unique user identification (IDs) and passwords to control access.
- Identification and authentication mechanisms stored in the system must be encrypted in accordance with FIPS standards.
- The Supplier must execute DoN and DoD policies regarding password expiration times and minimum password lengths.

- The Supplier must be able to support Secured Network Communications. (i.e. SSL, PKI).
- The Supplier must provide the processes, procedures, and a security administrator to maintain unique user identification and password control access into midrange environments, not specific DBMS or applications.
- The Supplier must implement a system where the user is responsible for maintaining and changing their password on a server in accordance with the security policy.
- The Supplier must process authorized requests to create, delete, or change a user ID from an authorized submitter.
- The Supplier must provide the avenue to receive and respond to user problems in the areas of sign-on difficulties, password resets, and Logon/Login/Sign-On assistance. Response times are outlined in the service level agreements.
- The Supplier must maintain control of all Administrator/root access to all network and server hardware including applicable disk storage devices such as EMC RAID arrays.

6. Customer Support Services

The Supplier must have Customer Support Services that provide request management through a Supplier liaison. The Supplier liaison must provide a communication focal point to facilitate all systems support and professional services. Client Service Management for the Essential Services also includes business hours operational support coverage, problem management, and change management processes.

a. Request Management

The Supplier must have Request Management Services that provide a communications liaison to facilitate rapid response to the Navy's requests. These services must include coordination to receive and process the Navy's requests for services. Examples include Platform Solution Design Services, Site Migration Services, Software Refresh Services, and Shared Services to accommodate ongoing Navy business needs or growth requirements. Requests may also address a temporary service requirement, a temporary service level requirement, or the implementation of a long-term requirement in which the Service Level Agreement must be revised.

Request Management Services Requirements are:

- The Supplier must have a process to receive and execute requests.
- The Supplier must provide oversight and coordination to understand request requirements to ensure deliverables and timeframes are met for the execution of the requests.
- The Supplier must mediate scheduling conflicts between program managers that have applications residing on the same server.

- The Supplier must provide regular communication of issues, concerns, and request schedules and attend application systems meetings when requested by the Navy Application Program Manager.

b. Continuous Hours Operational Support Coverage

The Supplier must be able to provide continuous hours of coverage by skilled staff to support all selected compute management packaged services. The Supplier must provide all systems management functions from the Supplier's monitoring location 24x7x365 and all other Supplier personnel required to provide the selected packaged solution services must be readily available 24x7 as necessary. If continuous support is not necessary, services can be adjusted based upon application requirements.

Continuous Hours of Operation Support requirements are:

- The Supplier will have skilled staff to support the midrange environment and all Enhanced Services.
- The Supplier must provide a monitoring location with on-site leveraged staff to monitor 24x7x365.

c. Change Management

The Supplier must have a Change Management process that controls changes to the midrange compute environment. The Supplier's Change Management process will allow for the proper planning, analyzing, testing, communicating, and scheduling of hardware, system software, and environmental changes. Any changes made to the application, server software and hardware, or the infrastructure must be briefed at the Navy Program Manager's Change Review Board (CRB).

Change Management Requirements are:

- The Supplier must participate in the program's CRB as they are scheduled.
- The Supplier must document and track scheduled changes and status. Configuration documentation is available upon request.
- The Supplier must manage dependency requirements for all change scheduling.
- The Supplier must assist Navy Program Managers in assessing the risk of proposed changes, including review of change complexity, dependencies, duration of the change, ease of recovery, potential impact, and feasibility of the proposed implementation date.
- The Supplier must evaluate application changes to ensure that there is adequate resources and capacity to support the application.
- The Supplier must research and test all proposed system software upgrades and patches.
- The Supplier must manage and brief the status of proposed changes according to established CRB processes.
- The Supplier must assist Navy Program Managers in coordinating required testing to enable the successful implementation of changes.

- The Supplier must have a process in place that addresses the severity of change requests. The Supplier and the Navy Program Manager will determine the criticality of the change to ensure it is addressed in a timely manner as defined in the SLA's.
- The Supplier and the Navy must establish a mediation process to address changes that affect the contract, service level agreements or resource requirements.
- The Supplier must coordinate with the CRB in scheduling maintenance downtime and testing.
- The Supplier should document any tuning actions. If OS files are modified, that action should be documented. Routine tuning does not need to be presented to the Change Review Board.

d. Problem Management

The Supplier must have a developed Problem Management process that details the actions to be taken in response to operational issues. This process should enable timely communication of the status and corrective actions. Problem resolution must be prioritized based on the severity of the problem. As part of the Problem Management process it may be necessary to bring the critical application back on-line before the root cause of a problem is determined. If a problem persists, then the Supplier must coordinate a time with the Navy Program Manager to determine the root cause of the problem while allowing the application to be off-line for a longer period of time. For non-critical applications more time can be taken to determine the root cause of a problem.

Problem Management Requirements are:

- The Supplier must maintain a Help Desk with a centralized phone number for reporting and resolving problems. The Supplier's Help Desk must interface with the NMCI Help Desk because the Navy has designated that trouble calls be reported to the NMCI Help Desk first.
- The Supplier must prepare and communicate with the Navy Program Manager impact statements documenting the cause of the problem, the efforts required to temporarily correct the problem, a root cause analysis, and any follow-up steps. In addition to notifying the Navy Program Manager of a problem, updated status of the problem resolution, and estimated completion times must be provided as well.
- The Supplier must escalate any problems exceeding a response threshold based on severity of the problem. Thresholds are outlined in the service level agreements (SLAs).
- The Supplier must assist the Navy technical support staff if problem resolution points to the Navy application instead of the operating system or infrastructure.
- The Supplier response times will be determined by the negotiated SLA's.
- The Supplier must provide a monthly report to the Navy Program Manager with the appropriate help desk statistics, trend analysis, and a brief summary of the problems experienced, the means in which they were resolved, and the time necessary to fix the problem.

- The Supplier must coordinate with the Navy Program Manager to determine if they need to test the application to evaluate corrective action. All configuration changes resulting from the problem resolution must be documented and relayed to the CRB.

7. Service-Level Management

The Supplier must provide Service Level Management Services through a communications liaison. The liaison must provide the avenue to understand and address the Navy's issues and concerns as well as be aware of the Navy's future plans, which would impact midrange services. The liaison will be the Navy's contact for reports and SLA issues and will work with the Navy's Program Managers to develop strategic and tactical plans for the hosted systems.

Service-Level Management Requirements are:

- The Supplier must provide oversight of Service Level requirements and monitor and escalate any issues as necessary to help meet required Service Level standards.
- The Supplier must provide regular communications (weekly) and participate in joint planning processes (if necessary) with the Navy Program Managers and application teams to integrate service level management issues with directions on tactical and strategic planning; and near-term and long-term initiatives.
- The Supplier must work with the Navy Program Managers to develop a yearly IT plan that addresses Navy Program Manager requirements and the needs of the systems being hosted. The plan should include the expected growth rate of the application's user base, storage requirements, software releases, resource needs, future application releases, etc.

a. Standard Service-Level Management Reviews and Reporting

This service provides quarterly Service Level Management Reporting and Reviews. The Supplier must provide reporting with data to measure conformance to the service levels on a quarterly basis. Additionally, the Supplier must provide application specific weekly change reports and quarterly trends reporting for all change metrics.

Standard Service-Level Management Reviews/Reporting Requirements are:

- The Supplier must provide standard quarterly reports that outline the Supplier's services against those delineated in the Service Level Agreements.
- The Supplier must conduct quarterly review meetings to discuss service level reporting information.
- The Supplier will provide at least one weekly report that describes change activity for the midrange systems to include description of change, system affected, date and time of change, duration of change, and status of change for approved changes.
- The Supplier will provide a quarterly report of change activity metrics that includes the number of changes, number of successful changes, missed change windows, and number of changes not meeting lead-time requirements.

8. Business Continuity

Business Continuity involves the planning and implementation of procedures that ensure critical business operations resume following a disaster and that they return to normal operations as soon as possible. Part of the process is determining which applications are critical and which are not, then deciding upon the time frames for recovery and site recovery necessary to meet the recovery needs. Site recovery options are discussed in the Recovery Site Requirements section of the Enterprise Foundation Services of this document. Business Continuity is also referred to as contingency planning, recovery planning, business resumption planning, or disaster recovery planning.

a. Documented Recovery Action Plan

The Supplier must maintain a plan for recovering the midrange operating system and related system software. The Supplier must work with the Navy Application Program Managers to define the appropriate software recovery plans. The plans can be tailored to the solution defining the backup schemas, critical components, and test plans based on the specific workload. The recovery plan provides the processes, and documentation covering tape backups, recovery, and disaster recovery.

Documented Recovery Action Plan Services Requirements are:

- The Supplier must maintain documented recovery procedures for restoring the operating system and related system if a disaster occurs.
- The Supplier must conduct an annual review of the midrange environment to determine whether the operating system data backup and off-site storage rotation schedules meet recoverability objectives.
- The Supplier must have documented hardware and software configuration data to ensure the system is recovered to the most current environment.

b. System Backup and Recovery

The Navy needs to have operational support and management processes that meet operating system and related application requirements for data availability, accessibility, and retention. This service allows all system software and related storage configuration to be recovered if an operational or hardware failure occurs. This service supplements the Business Continuity Services that allow recovery if a disaster occurs. All backup media and the information on the media relating to the application or application database is the property of the Navy.

System Backup and Recovery Requirements are:

- The Supplier must implement backup software that monitors the backups via log files and reports any files that were not successfully backed-up.
- The Supplier must adhere to the documented backup plan to ensure that a minimum of one backup copy is maintained for each critical file. The normal backup schedule is where backups are performed daily 6 times a week and a full backup is

performed on Saturday or Sunday. Additionally a full monthly and end of year backup are performed. Unless increased by the Navy Program Manger the minimum retention requirements for backups are:

- Daily incremental backups
- Weekly full backups must be stored for 2 months
- Monthly full backups must be stored for 12 months
- Annual full backups must be stored for 5 years.

- The Supplier must implement backup software that verifies backed-up files by reading what was written.
- The Supplier must implement backup software that is able to perform unattended automatic backups of all systems.
- The Supplier must test full system restoration of the systems, including hardware, software and processes annually at a minimum or as specified by the Navy Program Manager. Results and lessons learned must be provided to the Navy Program Manager.
- The Supplier must have a process in place to facilitate requests for recovery of application specific files. The restoration times for each hosted application will be addressed in the application's SLA.
- The Supplier must monitor, verify, and escalate issues as necessary for operating systems and related application software backups and authorized restores.
- The Supplier must manage operational support processes for performing operating system and related application software recoveries as required in resolving software and hardware problems.
- The Supplier must adjust data backup and restore plans as new components are added to the system or availability requirements change.
- The Supplier must maintain the tape library to ensure the availability of the media and storage location to include scratch and foreign tapes.
- The Supplier must provide and maintain media including media reliability evaluation and aging and replacement processes.
- The Supplier must dispose of old backup medium in accordance with DoD and DoN policies.
- The Supplier must store the on-site backup medium in a separate space as the systems that are being backed up to ensure the safety of the medium in case of a disaster.
- The Supplier must transfer magnetically stored media to a new medium every three years to prevent degradation.
- At the conclusion of the contract, or if the contract is terminated for cause, the Supplier must deliver all application specific backup media and corresponding documentation to the Navy Program Manager.
- The Supplier must monitor and manage the SAN or NAS network if used.
- The SAN or NAS network can only be used to backup military/government applications. No civilian applications can utilize the same network to perform backups. The entire SAN or NAS network and system will be protected at the same level as the highest security classification of the information that it is backing up.

- Each tape must be protected in accordance with the highest security classification of any information on the tape. For example, if a tape contains information that is sensitive, but unclassified (SBU), and the tape also contains information from another application that is unclassified, the tape must be treated as SBU. Any confidential information on a tape makes the entire tape confidential.

c. Off-Site Tape Services

The Supplier must provide the processes necessary to ensure a copy of the operating environment (operating system, system software and application software) is stored in a secure, off-site location.

Off-Site Tape Services Requirements are:

- The Supplier must prepare tapes for shipment to the off-site tape vault.
- The Supplier must provide off-site vault storage for backup and recovery media.
- The Supplier must provide transportation of backup and recovery media to and from the vault.
- The Supplier must provide a mechanism for specifying which tapes are to be returned from the vault.
- The Supplier must audit the off-site storage location at least annually.
- The Supplier must ensure that each tape is properly documented and labeled.
- The Supplier must ensure that at a minimum the full weekly backups are stored offsite.

d. Disaster Recovery Test Service

The Supplier must be able to provide full testing for the documented recovery action plan. Testing verifies that the Disaster Recovery Plan meets the Navy's Application Program Manager's requirements. It also can be used to evaluate how well the recovery plan integrates with the Supplier's other service providers to provide timely recovery from a disaster. At the Navy's discretion, network personnel, the application team, and some set of the user base can be involved to test the recovered environment along with the Supplier's staff. After each test is complete, the Supplier must identify any deficiencies encountered and enhance the plan if required to meet the Application Program Manager's recovery objectives.

Disaster Recovery Test Service Requirements are:

- The Supplier must conduct annual recovery testing based on the recovery option chosen by the Navy's Program Manager for the specific application.
- The Supplier must be able to restore the operating environment from the data backups.
- The Supplier must verify and test operating environment functionality.
- The Supplier must coordinate with the application team and user base for testing time, as required.

- The Supplier must provide annual drill reports to include recommendations on procedural changes that can make data restoration time frames more cost-effective while meeting realistic recovery requirements.

e. Recovery Site Requirements

Recovery sites are necessary to meet the Navy's business continuity needs. The Supplier must offer three levels of recovery facilities – shell-site, warm-site, and hot-site. These options are linked to the Business Continuity Services described in the Essential, Enhanced, and Premier Packages. The shell-site option is mainly targeted for the Essential Package, which provides only off-site tape storage. The warm-site option most closely matches the Enhanced Package and the hot-site option aligns with the high availability services provided in the Premier Package.

This section is an extension of the Business Continuity Service requirements and is not meant as a replacement for any other requirements in this document. All other requirements for the hosted applications are implied in this section.

Hosted application systems will be designated as requiring one of three levels of recovery facilities. These are defined as shell-site, warm-site, and hot-site recovery sites. The Supplier must be able to provide each of these facilities. The Supplier must also be able to accommodate changes to an application system's recovery facility needs.

To reiterate the requirement is that the Supplier be able to provide these sites (through contracts, existing partnership arrangements, etc…), not that the Supplier has to actually has to own, staff, or manage these sites on a full time basis. Service level agreements will determine if a hot site is needed, and whether it will have to be staffed for contingency purposes.

Shell-Site Recovery Facility requirements:

- Must meet all the General Facility requirements excluding the Structural Requirements.
- No hardware is available to support the applications that are running.
- The facility used must not be in the same physical location as the production facility.
- Shell-Site recovery testing for critical applications must be done at least annually.
- A third party may provide the facility and hardware.
- Documented procedures for redirecting applications to the Shell-Site Recovery Facility must be developed and maintained by the Supplier.
- Warm-Site Recovery Facility requirements:
- Must meet all the General Facility requirements.
- The facility used must not be in the same physical location as the production facility.
- A third party may provide the facility and hardware.
- For designated applications and systems the hardware equivalent to the production environment is available in the warm-site facility.
- Warm-Site recovery testing for critical applications must be done at least annually.

- Documented procedures for redirecting applications to the Warm-Site Recovery Facility must be developed and maintained by the Supplier.
- Hot-Site Recovery Facility requirements:
- Must meet all the General Facility requirements.
- The facility used must not be in the same physical location as the production facility.
- The facility and hardware must be maintained in a standby operating environment or as part of a high-availability server implementation located in two physical locations.
- Hot-Site recovery testing must be done at least annually.
- Documented procedures for redirecting applications to the Hot-Site Recovery Facility must be developed and maintained by the Supplier.

9. Facilities - General Requirements

Facilities are defined in this section as the physical locations of the hardware. The services addressed in this section include but are not limited to electrical power, HVAC controls, structural characteristics of the areas where the hardware is located and security as it concerns the physical access to the areas where the hardware is located.

All Facilities must comply with DoN and DoD requirements**.**

a. Electrical Power

- The facility must have a clean energy source. Power fluctuations must not affect the equipment.
- In data centers, emergency power-off switches that shut off all power supplies must be installed and be readily accessible with posted notices showing their location. The Supplier must monitor the emergency power-off switches continuously.
- Backup electrical facilities (e.g., generators) are needed to ensure long term uninterrupted power. The facility must have n + 1 generators.
- Backup electrical facilities must be tested annually at a minimum.
- Each server must have access to a secondary power source.
- In the event of a power failure, Uninterruptible Power Supply (UPS) systems must be configured and tested to ensure safe operations of critical hardware for a minimum of 30 minutes and to carry the load until automatic switching to the backup power supply takes place.

b. HVAC and Climate Controls

- Facilities must be climate controlled and have environmental conditions conducive to multiple computer systems.
- The air conditioning unit must be included in the fire suppression system, so in case of a fire the A/C shuts off.

- Sensors and alarms must be installed in data centers to monitor the environment surrounding the equipment to ensure that climate controls remain within the levels specified by equipment design.
- The Supplier must monitor environmental controls and take actions based on detected problems or issues.
- Reports of the climate control systems must be generated monthly at a minimum.
- The computer room should have positive air pressure.
- Fire Suppression
- The data center must have its own alarm systems.
- Fire Suppression must be a pre-action / dry pipe sprinkle system and a gaseous system such as the replacement agent to Halon 1301, called FM-200. These systems must meet the National Fire Protect Act 75 as well as comply with most NAVFAC requirements to ensure the overall system adheres to commercially acceptable standards.
- The facility must ensure that it has working smoke and heat detectors.
- Computer supplies (for example paper) must be stored in a separate location away from the computer equipment to minimize risk of fire damage.

c. Structural

- Drop ceilings must include smoke, heat and water sensors.
- The facility must have a raised floor to support connections and airflow.
- The facility must have a loading ramp or easy access for loading equipment.
- Raised floor loading capacities must be a minimum of 150 lbs. / sq. ft.
- Raised floor must support a minimum-rolling load of 600 lbs (272 kg.) over the entire floor.
- The minimum floor loading capacities for the mechanical, electrical and battery room must be 400 lbs / sq. ft.
- Exterior walls should be able to withstand wind loads of 115 mph (185 kph). This is equivalent to a ‘class 3’ hurricane.
- Exterior envelope wall and roof deck composites should include a vapor barrier.
- No windows or curtain walls will abut the area where servers are located.
- Servers must not be housed in areas subject to flooding or water infiltration through walls, floors or ceiling.
- Walls separating critical mechanical and electrical equipment rooms must extend from the floor slab to the bottom of the roof or floor deck above and must be constructed with a minimum of a 2-hour fire rated assembly.
- Walls surrounding mission critical equipment in the data center areas must be constructed with a minimum of a 1-hour fire rated assembly.
- Walls surrounding magnetic tape and other media storage must extend from the floor slab to the bottom of the roof or floor deck above.
- Walls surrounding magnetic tape and other media storage must be constructed with a minimum of a 2-hour fire rated assembly.

- Blueprints must be available with markings for the following:
  - Power Supply
  - Fire Suppression
  - Access Points
  - Point of Presence to outside networks (PoP)
- The Supplier must comply with all Uniform Federal Accessibility Standards (UFAS) and must incorporate the American Disabilities Act (ADA) in its structural designs.

d. WAN/BAN/LAN Connectivity

- The Supplier must provide the service to connect geographically separated Navy and Marine Corps users/devices/printers. The Supplier must provide connection to external networks, for example:
  - Non-Secure IP Router Network (NIPRNET)
  - Secure IP Router Network (SIPRNET)
  - FTS-2001
  - Defense Research Engineering Network (DREN)
  - Defense Switched Network (DSN)
  - Public Switched Telephone Network (PSTN)
  - NMCI provided wide area transport services (commercial/DISA)
  - The Internet
- The Supplier must provide service to interconnect geographically co-located Navy and Marine Corps LANs and BAN attached devices.
- The data center's network must conform to DoD and DoN Internet and Intranet security policies.

e. Facility Physical Security

- Data center personnel are required to have picture identification badges.
- The Supplier must adhere to the personnel guidelines outlined in section 1.1.4 Contractor Specific Internal Information Guidelines of the N/MCI Contract N00024-00-D-6000 Attachment 4 Security Requirements document. Section 1.1.4 of the N/MCI Contract N00024-00-D-6000 can be found in Appendix A.
- Visitors must sign in and be escorted into and out of the facility to provide an audit log.
- A log of physical access to controlled areas must be kept.
- A list of individuals authorized to grant physical access to controlled areas must be maintained.
- A list of individuals granted physical access to controlled areas must be maintained.
- Access to secure areas must be protected by an electronic access control system.

- Access to data center equipment must be physically restricted to authorized personnel by locating the equipment in a closed area.
- The facility must have surveillance covering the entire server area 24x7x365.
- Detection devices or true floor to ceiling data center perimeter walls must be installed to prevent unauthorized access attempts.
- Physical security must implement multiple access control points with access controls to restrict access to authorized parties only (i.e. Tape Librarians should only have access to the tape library.)
- Attempts to gain unauthorized access to secured areas must be reported on a monthly basis.

10. Shared Services

Shared services are described as the use of shared servers and disk arrays that are utilized by multiple application systems. The Supplier must be able to use a strategy of leveraging its infrastructure to support the Navy's current and future business needs. Shared services should be used to help the Navy reduce its overall operations costs by making efficient use of available resources. Shared services should be available on a case-by-case basis determined by the supported applications requirements.

The application requirements are defined in the review of the application that is performed as part of the Midrange Site Transition Services for the application. As part of the review the Supplier and the Navy Program Manager will determine if shared services are appropriate for the application and if the use of shared services will enhance the performance, price and availability of the application in the hosted environment.

a. Shared Services – Disk

Shared Disk options include the use of current technology providing state-of-the-art speed of access to midrange disk components. The advantages of using Shared Disks are the availability of capacity on demand, application availability and economies of scale for large applications and databases.

b. Shared Services – Platform

Shared Platform Services are the use of state-of-the-art midrange servers that are able to support multiple application environments with the ability to reconfigure and reallocate server resources on the fly. These platforms may be implemented by the Supplier as a means of providing on-demand processing capacity and flexibility for the hosted application systems. Shared platform usage should be based on specific Navy application systems resource requirements as defined in the application requirements, selected SLA's and audit results.

11. Essential Services – Optional Service Upgrades

The Supplier must provide for service upgrades described in this section. The upgrades can be selected at an additional charge to expand the range of services provided in the Essential Package based on application-specific requirements.

Upgrade – No Upgrades defined for the Essential Services

a. Essential Services –Optional Service Adjustments

The Supplier must be able to adjust the service offerings for the Essential Services. These service adjustments can be selected to reduce the range of services provided in the Essential Services Package based on application-specific requirements.

Adjustment – No Documented Recovery Action Plan

This adjustment removes the Documented Recovery Action Plan Services from the Essential Services Package.

Adjustment – No Disaster Recovery Test Service

This adjustment removes the Disaster Recovery Test Services from the Essential Services Package.

Adjustment – Business Hours Operational Support Coverage

The Supplier should be able to adjust the 24X7 coverage provided in the Essential Services Package and reduce the level of coverage to support the times users are accessing the system. The support hours needed may be 8 or 16 consecutive hours per day across five consecutive business days (Monday – Friday) or seven business days (Monday – Sunday) depending on the location of the user base of the application.

B. ENHANCED BASE PACKAGE SYSTEM SUPPORT AREAS

The systems support services described in this section encompass the Enhanced Packaged Systems Support Services. These services can be expanded with the selection of upgrades for an additional fee or reduced with the selection of adjustments that reduce pricing.

The services provided in the Enhanced Package are designed for dynamic, growing applications that are critical to the Navy's business enterprise.

1. Systems Management

The Supplier must provide Systems Management Services that include all services defined in the Essential Package plus system DBMS monitoring and printer definition and queue management.

a. System DBMS Monitoring

Administration and support of a DBMS is divided into two separate areas of responsibility: System Database Support and Application Database Support. System Database Support and Application Database Support functions are differentiated as follows:

- System Database Administration is responsible for managing global DBMS resources that perform functions that require DBMS owner userid authority or functions required to provide overall system integrity for the database (e.g., installation of the DBMS Server software, runtime procedures and parameters for the database instance, creating users and access rights, creating DBMS tablespaces, creating and maintaining rollback and redo logs, etc).
- Application Database Administration is responsible for managing objects within the database (e.g., the Table definitions, indexes, views, procedures etc).

Throughout this document, anytime DBMS requirements are discussed they are directed toward System Database Support and not Application Database Support, which is the responsibility of the program manager.

The Supplier must be able to support Database Management System (DBMS) Monitoring Services that provide the required operational support to monitor the Navy's DBMS environments.

System DBMS Monitoring requirements are:

- The Supplier must monitor DBMS throughput and performance.
- The Supplier must monitor DBMS availability.
- The Supplier must provide a monthly report for DBMS availability as part of the service-level management services.
- The Supplier must monitor to detect potential DBMS problems.
- The Supplier must monitor databases for space utilization, database performance, and other specific database criteria such as dead locks.

Note: These support services do not include the services of an application database administrator, but rather the services to maintain the system level components of the DBMS system.

b. Printer Definition and Queue Management

The Supplier must provide Printer Definition and Queue Management Services that provide the support and processes required to define printers to a midrange system and to manage print queues on a midrange system to resolve problems in the queues through purging and resetting print jobs and queues. Problems are reviewed and actions taken as required in accordance with the problem management procedure. Manual manipulation of print jobs within the queue is not included.

Printer Definition and Queue Management requirements are:

- The Supplier must have a defined printer definition process.
- The Supplier must manage throughput of print queues.
- The Supplier must install and test the printers that are located in the hosted environment.
- The Supplier must resolve problems, including resetting or purging jobs, as needed.
- The Supplier must ensure that applications that print to network printers have the necessary connectivity to the network and that the printer is properly set up on the server. The Supplier must work with NMCI and program management staff to resolve connectivity and reach back problems.

2. Software Management

The Supplier must provide Software Configuration Management Services that include all services defined in the Essential Services Package plus system DBMS support services.

a. System Database (DBMS) Support Services

The Supplier must provide System DBMS Support Services that include the processes to plan, install and maintain the required DBMS operating environment to support DBMS software. These support services do not include the services of an application database administrator, but rather includes those services required to maintain the system level components of the DBMS system.

System DBMS Support Service requirements are:

- The Supplier must configure, install, and test DBMS system environment.
- The Supplier must maintain, install, and test DBMS upgrades and patches. All DBMS changes must be presented to the Change Review Board.

- The Supplier must, in coordination with Navy Program Managers, maintain documentation of DBMS configurations including application software release levels, configurations, patches, etc.
- The Supplier must maintain documentation of all changes approved by the Change Review Board including date approved, change summary and date change applied.
- The Supplier must make all documentation available to the Navy upon request.
- The Supplier must create, maintain, and execute DBMS system start-up/shutdown scripts and processes.
- The Supplier must maintain and configure DBMS system disk including slicing and placing.
- The Supplier must create and maintain DBMS files, DBMS tablespace, and application tablespaces.
- The Supplier must verify effectiveness of changes on DBMS files and tablespaces utilizing an approved test plan.
- The Supplier must perform backup and recovery of DBMS system files and tablespaces, as well as the database application itself. Backup schedules and storage requirements are outlined in backup section of the Essential services.
- The Supplier must maintain DBMS Backup/Recovery and Disaster Recovery Procedures and Documentation.
- The Supplier must manage and if necessary modify DBMS file and DBMS tablespace characteristics.
- The Supplier must participate in design reviews and project meetings to provide technical guidance for DBMS related issues.
- The Supplier must work with the Navy Application Program Managers and Navy Technical Application Support Staff to resolve DBMS performance related issues.
- The Supplier must maintain security and access to the DBMS and its associated files.
- DBMS software refresh provides the same services outlined in the software refresh portion of the Essential package.
- The Supplier must work with the Navy's DBA to install application updates or patches.

3. Workload Management

The Supplier must be able to provide Workload Management Services that include support for Batch Scheduling and Batch Monitoring to determine whether production batch cycles are completed in required time frames.

a. Batch Scheduling Services

The Supplier must be able to provide Batch Scheduling Services that involve activities associated with defining and maintaining the execution requirements of an application's batch processing that is scheduled under the system's automated

scheduling product. The objective of production batch scheduling is that all pre-defined application cycles execute in the proper sequence with cycle completion scheduled realistically within the defined processing windows.

Batch Scheduling Services requirements are:

- The Supplier must maintain the job-scheduling database for the automated scheduling product.
- The Supplier must perform day-to-day maintenance and operational support of the scheduling system.
- The Supplier must perform additions, changes, or deletions to the scheduled batch workload as requested by authorized personnel.
- The Supplier must assist the Navy Application Program Managers and Navy Technical Support Staff in performing batch scheduling or cycle flow problem determination.

b. Batch Monitoring Services

The Supplier must provide Batch Monitoring services to support processes necessary to monitor the application batch cycle. If abnormal termination or a restart occurs, the scheduled batch processing will be executed based on pre-defined instructions or the issue will be escalated to the Navy's Application Team as necessary.

Batch Monitoring Services requirements are:

- The Supplier must monitor resource availability, abnormal termination, and cycle start and end times for scheduled batch processing. The Supplier must provide monthly of reports of batch processing statistics to the Navy Program Manager.
- The Supplier must perform and/or assist the application team in performing production batch restarts and reruns.
- The Supplier must assist the Navy Technical Support Staff in resolving abnormal termination because of system abnormalities.

Note: The Batch Monitoring Services do not include monitoring of the execution of user-submitted jobs.

4. Application Security and Resource Controls

The Supplier must be able to install any required software tools and set up access parameters and ongoing support required to create and maintain application resource controls for the midrange environment in accordance with the Navy's Application Team requirements.

Application Security and Resource Controls requirements are:

- The Supplier must provide processes to secure application files according to DoN, DoD, and NMCI security requirements.
- The Supplier must manage user access to the applications including the processes and procedures necessary for Adding, Updating and Deleting user access.

- The Supplier must have a process in place to receive and respond to user problems in the areas of file access difficulties and security violations.

5. Production Promotion

Production Promotion Services include change control services and software for managing the promotion of source and object code for developed programs or applications from test to production environments. This service is designed to make the Supplier responsible for migrating changes into production alleviating the need for Navy Program Managers to perform such tasks. As such the program manager will not need to gain the assistance of the Supplier to gain root access to install an update. The program manager will give the update to the Supplier and the Supplier take all of the steps necessary to install the application update.

The Supplier must provide Production Promotion Services for ongoing maintenance of the hosted applications. These services incorporate procedures for promoting application software changes and application file changes made by the Navy's technical staff into the hosted application's production environment.

Production Promotion Support Services Requirements:

- The Supplier must provide a change control process for source and object code promotion.
- The Supplier should provide version control for source and object code.
- The Supplier must manage the promotion of source and object code from test to model office to production files or server environments.

6. Customer Support Services

The Supplier must provide Customer Support Services that include request management, change management, problem management, and service-level management as they affect the midrange environment. Besides the services provided in the Essential Services Package, the Enhanced Package provides regional coordination of requests.

a. Request Management – Multi-Site Coordination Services

The Supplier must be able to provide Enhanced Request Management Services that include the coordination of receiving and processing Navy requests for services in a single location as provided in the Essential Services Package, but also regional request coordination. The Supplier must be able to provide request coordination via a single client liaison across multiple regional processing environments that are under the Supplier's control.

This service should integrate software and hardware refresh requests, coordinate scheduling, and provide regional consistency while meeting the Navy's application-specific business requirements. Regional coordination in this context is across multiple sites. When requests requiring this level of coordination are received, the

Supplier's request management processes should provide regional communications to coordinate and execute the request among all required locations.

Request Management – Multi-Site Coordination Services requirements are:

- The Supplier will review all requests to determine and understand potential regional requirements and present the findings to the Change Review Board.
- The Supplier will monitor request status across all impacted regional sites to determine whether deliverables and time frames are met among all environments throughout the region as required. The Supplier will brief the request status to the program manager on a weekly basis.
- The Supplier will coordinate the scheduling of actions resulting from the request across affected sites.

7. Enhanced Service – Optional Service Upgrades

The service upgrades can be selected to expand the range of services provided in the Enhanced Services Package based on client-specific requirements.

a. Upgrade – Custom Product Support

The Supplier will integrate and support a completely customized set of products as defined by the Navy Application Program Managers and the Supplier's Technology Advocate. This set of products should be fully integrated into the operating platform package for installation. Please see the Premier Services Package definition for a detailed list of services.

b. Upgrade – Local High-Availability Support

The Supplier will support High-Availability Services that provide processes and support for redundant server and storage environments that are clustered together in the same physical site. Please see the Premier Services Package for a detailed list of services included.

c. Upgrade – Custom Service Level Reviews and Reporting

The Supplier will to provide customized service level reviews and reporting. Please see the Premier Services Package for a detailed list of services included.

d. Enhanced Services – Optional Service Adjustments

These service adjustments can be selected to reduce the range of services provided in the Enhanced Services Package based on application-specific requirements.

Adjustment – No Printer Definition and Queue Management

The Supplier will remove the Printer Definition and Queue Management Service of Systems Management Services from the Enhanced Services Package.

Adjustment – No Workload Management

The Supplier will remove all Workload Management Services from the Enhanced Services Package. This includes removing support for batch job and cycle scheduling as well as monitoring scheduled batch processing.

Adjustment – No Batch Scheduling

The Supplier will remove the Batch Scheduling Service of Workload Management Services from the Enhanced Services Package. The Operational Monitoring Service for scheduled Batch Processing is not affected.

Adjustment – No System Database (DBMS) Support

The Supplier will remove the System Database (DBMS) Support Service for Software Configuration Management and all monitoring of DBMS from the Enhanced Services Package.

Adjustment – No Production Promotion

This adjustment removes the Production Promotion Service of Workload Management Services from the Enhanced Services Package.

C. PREMIER BASE PACKAGE SYSTEM SUPPORT AREAS

The Premier Services Package is designed for Navy's most mission-critical systems that require a customized infrastructure design, build, and operation because of the business application complexity, diversity, and variety. In addition to the services provided in the Enhanced Services Package, the Premier Services Package provides support for more complex software and hardware configurations to provide high availability for business critical processing requirements.

1. Systems Management

Systems Management is the process of analyzing, evaluating, and reviewing the compute operation to verify that operational requirements are met. The range of services includes all services defined in the Enhanced Services Package plus Application Monitoring and advanced web site monitoring.

a. Application Monitoring

The Application Monitoring Service provides the Navy with proactive automation and monitoring that result in more stable, functional applications that meet Navy and operational requirements. Application monitoring involves more than the monitoring of application resources. Application monitoring can involve monitoring distinct functions within the application for input/output speed, checking for looping/hung processes, analyzing application usage patterns (which options or branches are used most often), and reviewing exception logs.

This service is designed for monitoring purposes only. Program manager cooperation may be required to interface monitoring agents with the application code.

Application Monitoring Services Requirements are:

- The Supplier must develop, install, and test specific application automation agents for use in application monitoring.
- The Supplier must configure, install, and test custom product automation agents.
- The Supplier must manage and monitor the application and/or custom product operational environment.
- The Supplier should monitor the application database to ensure the database is responding to requests if applicable.

b. Web Site Monitoring

The Web Site Monitoring Service provides the Navy with automated monitoring of web sites to ensure there are no broken links in the Web Site. A broken link is defined as a hyperlink from one web page to another that is no longer available.

Web Site Monitoring Services Requirements are:

- The Supplier should monitor identified web pages for broken links on a periodic basis as defined by the Navy Program Manager.
- The Supplier should provide results of broken links to the Navy Program Manager.

2. Software Management

Software Configuration Management Services provides for the installation, maintenance, documentation and upgrading of midrange environments. The range of services includes all services defined in the Essential and Enhanced Services Packages plus Custom Product Support and Local High-Availability Support Services.

a. Custom Product Support

The Supplier must integrate and support a completely customized set of products (operating systems, network devices, server hardware, etc.) as agreed upon by the Navy and the Supplier. Custom products in this context refer to support for nonstandard software or hardware that is utilized by the application. Operating systems such as Linux or BSD UNIX are nonstandard, and the contractor may have to hire additional personnel to support the software. Custom support does not refer to the application itself.

Custom Product Support Services Requirements are:

- The Supplier must support a custom-designed solution of system-related vendor products selected by the Navy and the Supplier.
- The Supplier must plan, install, integrate, and upgrade the custom product set.
- The Supplier must resolve problems, including problem determination, interface, and escalation with third-party suppliers, for the custom product set.
- The Supplier must install corrective and preventive maintenance to custom product sets.
- The Supplier must conduct inventory, track, and document the custom product set components and changes.
- The Supplier must provide software refreshes to allow early adoption or to maintain currency to current software versions of the custom product. Software refresh may not be applicable in some cases where the custom product is being used because of hard coded dependencies specific to a particular version.

b. Local High-Availability Software Support

Local High-Availability Software Support Services provide the processes and support staff to support system software required to provide redundant server and storage configurations clustered together in the same physical site.

Local High-Availability Support Services Requirements are:

- The Supplier must install and maintain the system software and related tools required to provide a midrange compute environment that meets availability requirements and removes single points of failure from the compute configuration.
- The Supplier must provide high-availability software expertise to manage and monitor the operational environment.
- The platform will support non-disruptive software maintenance to both the system software and the application.

3. Hardware Configuration Management

The Hardware Configuration Management of the Premier Services Package includes the processes and procedures for the installation, upgrade, coordination and oversight of midrange high-availability environments. The range of services includes all services defined in the Enhanced Package plus hardware refreshes as required to maintain state-of-the-art high availability configurations.

a. Local High-Availability Hardware Support

Local High-Availability Hardware Support services provide the processes and support for redundant server and storage configurations that are clustered together in the same physical site to support continuous availability requirements.

Local High-Availability Hardware Support Services Requirements are:

- The Supplier must manage platform solution configuration requirements to meet availability requirements and remove single points of failure from the compute configuration.
- The Supplier must provide subject-matter expertise to manage and monitor the operational environment.
- The Supplier must coordinate with vendors to provide non-disruptive maintenance processes.
- The Supplier must provide the capability to dynamically reconfigure resources to support applications experiencing high demand.

4. Customer Support Service

The Supplier must be able to provide Customer Support Services that include a more extensive range of Change and Problem Management Services. The complete set of services that are provided encompass all services defined in the Enhanced Package and additional client-specific change and service reviews.

a. Request Management – Global Coordination

Premier Request Management Services include not only the coordination of receiving and processing Navy requests for services within geographic regions as provided in the Enhanced Services Package, but also global request coordination. The Premier Services Package must provide request coordination via a single Supplier client liaison across all global processing environments. This service integrates such services as software and hardware refresh requests, coordinates scheduling, and provides global consistency while still meeting client-specific business requirements. When requests requiring this level of coordination are received, Supplier request management processes provide the global communication to coordinate and execute the request among all required locations.

Request Management – Global Coordination Services Requirements are:

- The Supplier must review all hardware and system software requests to determine and understand potential global requirements.
- The Supplier must communicate and monitor the status of the request across all impacted global sites to ensure deliverables and time frames are met among all global environments as required.

b. Custom Service Reviews and Reporting

Custom Service Reviews and Reporting includes additions to Standard Service-Level Management Reviews and Reporting. Navy-specific service level reporting must be available and customized to address unique reporting requirements. The review and reporting services for change and problem management can be customized to meet application specific requirements. More frequent problem and change management review services that encompass weekly Navy-specific problem review meetings and daily service review meetings for all problem metrics are also provided.

5. Premier Services – Optional Service Upgrades

The service upgrades can be selected to expand the range of services provided in the Premier Package based on program-specific requirements. There is an additional charge associated with each service upgrade.

a. Upgrade – Remote High-Availability Support Services

The Supplier must have Remote High-Availability Support Services that provide the processes and support for redundant server and storage configurations that are located in geographically distributed physical sites either via hardware and/or software tools to support specified availability requirements. Besides the protection provided by a local high-availability solution, a remote high-availability configuration provides business continuity if the local operating site is incapacitated.

The components of a remote high-availability configuration include remotely clustered platform configurations and remote mirrored disk storage configurations. When redundant sites are requested, the Supplier and the solution vendors perform a risk/cost/benefit analysis for Navy approval. Eliminating single points of failure helps prevent interruptions in service because of discrete hardware and software failures.

Remote High-Availability Support Services Requirements are:

- The Supplier must design and implement a configuration to meet availability requirements and remove single points of failure from the compute configuration.
- The Supplier must provide subject-matter expertise to manage and monitor the environment as defined by the services selected for the application.
- The Supplier must coordinate vendors to provide non-disruptive maintenance processes to ensure the availability of the hardware components of the compute configuration.
- The platform configuration must allow non-disruptive system and application software maintenance to ensure the availability of the software components of the compute configuration.

b. Premier Services – Optional Service Adjustments

These service adjustments can be selected to reduce the range of services provided in the Premier Package based on client-specific requirements. There is a price reduction associated with each service adjustment.

Adjustment – No High-Availability Support

This adjustment removes local High-Availability Services from the Premier Package. This includes removal of local High-Availability Services that provide the processes and support staff to support redundant server and storage configurations that are clustered together in the same physical site either via hardware and/or software tools to support continuous availability requirements.

Adjustment – No Request Management – Global Coordination Support

This adjustment removes Request Management – Global Coordination Services from the Premier Package.

6. Contract Termination

At the conclusion of the contract, the Supplier must assist the Navy Program Manager and any third party contractor in migrating the application to a new environment. This includes allowing a third party contractor access to the servers to evaluate the applications. The Supplier must perform the following actions upon the completion of the contract:

- Transfer the application or groups of applications to a suitable media for transport to the new environment.

- Provide software and hardware configuration information.
- The Supplier must provide audit information to assist any third party organization in gathering data necessary to migrate the application.
- The Supplier must turn over all application related backup disks.
- The Supplier must purge all application data from their systems in accordance with DoD and DoN regulations.
- The Supplier must provide all required end-of-month reports and documentation.

7. Acknowledgements

The author would like to acknowledge the assistance of Scott Price and Joseph Vickery from EDS. Their contributions throughout this paper, especially in the facilities portion of this paper, were invaluable.

D. NMCI CONTRACT (APPENDIX A):

N/MCI Contract N00024-00-D-6000 Attachment 4 Security Requirements Section 1.1.4

1.1.4 Contractor Specific Internal Information Guidelines

1.1.4.1 Classified (DoD) Information Support

The highest classification level of information required in connection with this procurement is TOP SECRET.

In accordance with the National Industrial Security Program Operating Manual, DoD 5220.M, the contractor shall possess or be able to possess a Facility Security Clearance equal to the highest level of classified information necessary to perform the tasks or services required on this contract.

Contractor personnel, whose duties require access to systems processing classified information, shall possess a security clearance at least equal to the highest degree of classification involved and shall have a validated need-to-know prior to beginning work on the classified system.

The sponsoring agency security requirements for classified systems shall be met by all contractor personnel accessing classified information, or contractor systems processing classified information.

The contractor shall perform internal assessments to determine position sensitivity and management controls necessary to prevent individuals from bypassing controls and processes, such as individual accountability requirements, separation of duties, access controls, and limitations on processing privileges at contractor facilities. These position sensitivity assessments will be forwarded to the Government for a determination of personnel suitability and requirements for individuals assigned to these positions in accordance with DRD3. Periodic re-evaluations of positions and suitability requirements will be necessary during the life of the contract as positions and assignments change.

The contractor shall conduct risk assessments, document the results, develop and maintain internal security plans. These plans shall describe how the contractor ensures the integrity, availability, and confidentiality of the information that it is operationally responsible to protect within the vendor’s facilities.

1.1.4.2 Sensitive Information Support (Non-classified)

Under current Federal guidelines, all officially held information is considered sensitive to some degree, and shall be appropriately protected by the contractor as specified in applicable IT Security Plans.

Types of sensitive information that will be found on DoN systems that the contractor shall have access to include, but are not limited to: Privacy Act information; proprietary information of other companies or contractors; resources protected by International Traffic in Arms Regulation (ITAR); technology restricted from foreign dissemination for competitive reasons; DoN administrative communications, including those of senior government officials; procurement or budget data; information on pending Equal employment Opportunity (EEO) cases; labor relations; legal actions; disciplinary actions; complaints; IT security pending cases; civil and criminal investigations; information not releasable under the Freedom of Information Act (FOIA) (e.g. payroll, personnel, and medical data).

The contractor shall perform internal assessments to determine position sensitivity and management controls necessary to prevent individuals from bypassing controls and processes, such as individual accountability requirements, separation of duties, access controls, and limitations on processing privileges at contractor facilities. These position sensitivity assessments will be forwarded to the Government for a determination of personnel suitability and requirements for individuals assigned to these positions. Periodic re-evaluations of positions and suitability requirements will be necessary during the life of the contract as positions and assignments change.

The contractor shall conduct risk assessments, document the results, develop and maintain internal security plans. These plans shall describe how the contractor will ensure the integrity, availability, and confidentiality of the information that is operationally responsible to protect within the vendor's facilities and at government facilities. For example the contractor shall ensure that foreign nationals within their corporate staff will not have access to NMCI data that is not releasable. A decision to accept any residual risk will be the responsibility of the DoN system owner and the DoN information owners. The contractors risk assessments and IT Security Plans shall be updated at least every three years or upon significant change to the functionality of the assets, network connectivity, or mission of the system, whichever comes first. If new or unanticipated threats or hazards are discovered by the contractor, or if existing safeguards have ceased to function effectively, the contractor shall update the risk assessments and IT Security Plans (within 30 working days) and shall make appropriate risk reduction Recommendations to the DoN system owner and the DoN information owners (within 5 working days).

1.1.4.3 Privacy And Security Safeguards

The contractor shall not publish or disclose in any manner, without written consent of the government, the details of any security safeguards designed, developed, or implemented by the contractor under this contract or existing at any DoN Center.

The contractor shall develop procedures and implementation plans to ensure that IT resources leaving the control of the assigned user (such as being reassigned, removed for repair, replaced, or upgraded) is cleared of all DoN data and sensitive application software by a technique approved by the government. For IT resources leaving DoN use,

applications acquired with a "site license" or "server license" shall be removed. Damaged IT storage media will be degaussed and destroyed.

To the extent required to carry out a program of inspection and audit to safeguard against threats and hazards to the confidentiality, integrity, and availability of government data, the contractor shall afford DoN access to contractor facilities, installations, technical capabilities, operations, documentation, records, databases, and personnel.

E. NAVSUP SERVICE LEVEL AGREEMENTS

Service level agreements have many formats depending upon how they are used. Internal SLAs between management and the IT department can be more informal because many of the procedural issues are stated elsewhere. SLAs involving external service providers need to be more formal.

SLAs serve as a mechanism to notify all parties of services that will be performed, performance expectations, responsibilities of all parties, penalties for non-performance, and SLA resolution procedures. SLAs also define the oversight and interaction between the program managers and the service provider.

SLAs are often used in conjunction with a Statement of Work (SOW), which provides the actual requirements. The SLAs provide the metrics to measure whether the requirements are being met. Most activities find it easier to keep the two documents separate, as many requirements will not have SLAs associated with them.

The following is the SLA template that NAVSUP will be utilizing:

Service Name: This is the name of the service category that is being measured (e.g., help desk support).

Service Description: This is a detailed discussion of the service that is to be performed. This represents the business function, process, or procedure that is to be measured.

Reason for Measuring: This section should provide the rational for this SLA. A valid justification prevents measuring for measurement sake. The results of the measurement should result in problem determination, lead to corrective action, and maintain the performance achieved by the corrective action. The SLAs should be linked to a strategic or tactical business concern.

Time Frame: This is the time period during which measurements are taken (e.g., 24x7x365, or from 0700-1900 Monday through Friday)

Scope: This section defines where the services apply (e.g., this applies to the system software only). This section also provides amplifying information such as categorization of problem calls (i.e., priority 1 equates to an emergency), and information necessary to ensure all parties understand the areas that are covered by the SLA. The scope also details areas not covered by the SLAs.

Performance Category: This section names sub-services that must be measured to determine the over-all efficacy of the service. There can be numerous performance categories associated with one SLA. The following subsections are associated with every performance category:

Performance Metric: This section describes the metric to measure performance.

Threshold Levels: This section describes the performance thresholds that must be met at the various service levels. There are generally more than one level of service. In the example that will be presented, three service levels will be used. Obviously as the thresholds become more difficult to meet, the costs of providing the service will rise.

Formula: The formula describes how the metric will be computed.

Assumptions: All assumptions should be stated in this section.

Contractor Responsibility: This section details the contractor's responsibilities in meeting the service level requirements.

Customer Responsibility: The program manager or the end-user's responsibilities are outlined in this section (e.g., a trouble call must be initiated before metrics covering the help desk can apply).
Frequency: This is the period of time over which measurements will be taken to determine SLA compliancy (e.g., monthly, quarterly). This usually equates to the periodicity of the reporting requirements.
Measurement Techniques: How will the metrics be gathered? This describes the procedures that will be used to collect the performance measurements.
Reports Required: This section details the reports required from the service provider to verify actual performance against SLA thresholds. It also details the periodicity requirements of the reports (e.g., Trouble Tickets – Monthly). The person reviewing the SLAs may have access to the report generating tool, and can manipulate the reports as needed. An example is if the reviewer has online access to the trouble tickets, that individual can do daily, weekly or monthly reports, at whatever level of abstraction is needed.
The specific reports required will be outlined in the Contractor Data Requirements List (CDRL), which is separate from this SLA. The CDRL will detail the format and content required, the frequency, distribution, and means of dissemination. The reports required will vary depending upon the type of application, the criticality of the application, monitoring tools used, funds available, and management needs. Typically daily reports are more technically oriented and are used by the CTR for verification; weekly or monthly reports are generally aggregate reports that provide service level summaries to management.
Person Responsible for Verification: This section details who will be reviewing the SLA measurements and determining compliancy. In the government, this person is usually the Contracting Technical Representative (CTR).

Escalation Procedures: This section describes actions to be taken when thresholds are exceeded, and who should be notified. For example if help desk response time is 15 minutes for a critical application, and 30 minutes have passed, who should be notified? This also includes situations where thresholds are violated on numerous occasions throughout the reporting period. This section also describes escalation procedures if the CTR and service provider cannot agree that a threshold violation has occurred.
Contractual Exceptions: This section describes the exceptions to the SLA. For example an emergency situation may require the service provider to violate a SLA threshold.
Penalties/Rewards: An SLA without penalties or rewards is nothing more than an agreement. SLAs must have a mechanism to enforce compliancy. This section describes what action will be taken if thresholds are violated, or if SLAs are met. It is important to identify minor and major thresholds to ensure that the service provider is taking action to correct the problems.

Service Name	SLA 1.0: Compute Service Availability
Service Description	Availability measures the capability of an end-user to access and fully utilize an application (according to specifications) over a period of time. Availability is usually expressed as a percentage of time that the system was available for use divided by the agreed upon hours of operation. The time period that an end-user cannot utilize the application is considered 'downtime'. Availability metrics are generally intended to be end-to-end, reflecting availability from the end users perspective. However, these SLAs only cover the host environment, so availability metrics will be restricted to the host environment only, and will not apply to the client piece or the connectivity from the client to the host environment firewall. Downtime can also be difficult to define. This SLA will concentrate on an application's opportunity to compute. The thresholds will contain metrics to ensure that the application has sufficient resources to operate to specifications. If the compute environment is not operating at a certain level of efficiency, the application performance suffers. As a result, if certain resource thresholds are not met, the period of time the resources do not meet the thresholds will count as downtime. Response time is another element of availability that must be addressed. The SLA is limited to the host environment, so application response time will be calculated from the time a server receives application input until it provides the correct output. It is necessary to develop a program that resides on the server in order to generate the information necessary to measure response time (this is often referred to as synthetic transactions). The program will test key application functionality at random times and measure the response time from when the input is initiated until the desired output is correctly received. Response times will apply to enhanced and premier services only. It is assumed that the government will develop the synthetic transaction software. Development of the program will be negotiated as a separate line item if the program wants the service

	provider to perform that function.
Reason for Measuring	Availability is a measure of quality. The program manager and the contractor need to constantly monitor the infrastructure, hardware and system software to measure the effectiveness of the hardware and software in supporting the application. Diligent monitoring will detect early signs of problems that may require maintenance action. The efficacy of the application support has direct business impacts. When the application is not available any business related to that application stops; opportunities are missed, business processes are impacted, and deadlines can be missed. The program manager must identify a target availability threshold and be able to justify expenses associated with it. This will involve determining the business impact of lost service. The contractor must evaluate the infrastructure to determine if it is possible to support the availability, or if redesign or additional redundant or high availability equipment is needed. The host environment cannot be designed, implemented, or managed unless an availability threshold is established.
Time Frame	Derived by the contracted number of support hours. The Default is 24x7x365. Scheduled maintenance time that is within the maintenance window, and does not exceed the agreed upon maintenance time frames will not be included in availability computations. Additionally, scheduled maintenance involving the application (i.e., granting root access to maintenance personnel to perform an upgrade) will not be considered down time. The Maximum "Available" time will be determined from the hours of support that were contracted. Example (1): Hours of Support = 24 x 7. The maximum "available" time in a 30 day month is 30 x 24 x 60 = 43,200 minutes. Example (2): Hours of Support = 9 x 5. The maximum "available" time in a month with 21 work days is:

	21 x 9 x 60 = 11,340 minutes.
Scope	This is an end-to-end metric from the host environment firewall to the application. It includes the hardware and the software for the firewall and server farm network, in addition to the hardware and software necessary to support the application. It does not apply to the application itself.
Performance Category	1.0 Host Environment Availability
Performance Metric	Availability is expressed as a percentage of the time that an application is fully functional divided by the total time encompassed in the support hours.
Threshold Levels	Availability thresholds are as follows: Essential Services: 99.50% Enhanced Services: 99.90% Premier Services: 99.95% In this SLA, availability is not only dependent upon the individual components that comprise the infrastructure (servers, network and firewall); it also addresses application and data availability from a security perspective. The following thresholds apply to resource utilization and network efficiency. If these thresholds are violated, then the application is considered 'down', and will count against availability: <u>Server Measures</u>: CPU Utilization: 75% sustained for over 1 hour. Not to exceed 90% for more than 2 polling cycles (5 minute intervals). Frequency of Failure: More than 3 service interruption in one day. Disk Utilization: 90% Disk Response Time: .25 second Disk Average Queue Length: 3 Disk I/O rate: 100 ms average Swap space availability: 90% of defined space Memory paging: 5 per second <u>Network Measures</u>: Data Delivery Rate: 99.95% LAN Latency (one way): 70 ms LAN Packet Collisions: More than 7% of packets transmitted (average based on a 1 hour interval). Bandwidth Availability: 85% of defined bandwidth

	Ethernet Segment Utilization: Less than 30% Security Related Measures: If application performance is degraded due to an intruder attack, virus, worm, or security breaches previously identified, the application is considered "down". This includes the time that the application is affected during efforts to correct the violation. New attacks that have no previous history or signature will not be counted as "down time" against availability as long as the attacks did not exploit vulnerabilities that were corrected by security patches that should have been installed. Application Response Time: Will be dependent upon the types of transactions that are being performed. If all transactions are similar, one threshold value can be determined (e.g., query requests must be generated and returned within 1 second). If the transaction response times vary considerably, the response thresholds should be specific to the transaction. In this SLA, response times are generated from synthetic transactions and are measured from the server only. All hardware errors affecting the application are considered 'downtime', and will be counted against availability.
Formula	Availability = (total uptime minutes) / (total uptime minutes + total downtime minutes) * 100
Assumptions	Downtime starts with the generation of a trouble ticket, or when the monitoring tools capture a threshold violation. Problems relating to the firewall, network, server or system software will count towards downtime. A review of the trouble tickets and monitoring software reports will verify that the downtime is properly assigned. Downtime attributed to application errors will not be included in the computation. Downtime that is a direct result of government actions will not be included in the computation. An example would be rebooting the system following an application update. Errors attributed to the client side portion of the compute environment will not be charged against reliability calculations.

Contractor Responsibility	Adopt and implement an industry-standard software solution for automatically polling and calculating compute service availability. Monitor compute services for earliest identification of outages. Take appropriate actions to correct deficiencies.
Customer Responsibility	The customer is responsible for prompt notification of any suspected compute service outages.
Frequency	Monitoring is conducted during scheduled support hours. Report frequency is monthly. Assigned government representatives will have real-time or near real-time access to monitoring software (read-only mode is acceptable).
Measurement Techniques	The server will be 'Pinged' from a management server every 5 minutes. Failure by the server to respond will start the service outage time. The time between the first 'Failed' Ping and the first successful Ping after repair will be reported as Downtime. Example: Server A polled at 10:40, 10:45 and 10:50 and does not respond to the 10:45 poll but does respond at 10:40 and the 10:50. This would be calculated as 5 minutes of downtime. Approved industry standard monitoring tools such as Tivoli® and Open View® will be used to monitor the server and network. Operating system logs will also be used to determine compliance. Threshold violations will be considered downtime. Each threshold specified will have to be evaluated to determine the period over which the measurement is determined. Unless otherwise specified, thresholds that specify averages will be computed over a 1-hour period. Other thresholds will normally be monitored in real-time, or near real time. "Down time" is considered when a threshold is violated for more than 5 minutes. The downtime will be reviewed and adjusted by a contractor representative to exclude all outages from maintenance windows or outside the scope of service: All planned outages

	All outages due to application failures Adjusted Compute Service Availability is then recalculated. The new formula would be as follows: Availability = (total uptime minutes – downtime outside of scope) / (total uptime minutes – downtime outside of scope + total downtime minutes) * 100 Example Calculation: Server contracted for 7 x 24 hour support. Two outages occurred during a month with 30 days: (1) 100 minute application outage and (2) a 360 minute system failure occurred for a total downtime of 460 minutes. Availability is reported as: Reliability = (43,200 – 100) / ((43,200 – 100) + 360) * 100 = 99.17%
Reports	1. Monitoring reports: Weekly, in addition to real-time/near real time viewing of the monitoring tools that will allow visibility to raw data. 2. Trouble tickets: Weekly
Person Responsible for Verification	The Contractor Technical Representative (CTR) will be responsible for reviewing the monitoring reports and trouble tickets to determine compliance with the SLAs.
Escalation Procedures	The CTR will be notified if the application is not accessible or functioning by the following time frames: Essential Service – after 30 minutes Enhanced Service – after 15 minutes Premier Service – after 10 minutes If there are any disagreements concerning whether downtime should be charged to the application, or the host environment, the CTR will make the decision. Disagreements can be escalated to the Contracting Officer Representative (COR).
Contractual Exceptions	Availability does not include scheduled maintenance downtime within the maintenance window.
Penalties/Rewards	Minor penalty: 10% of monthly rate • Threshold values exceed agreed upon rates. Major violation: 25% monthly rate • More than 3 minor penalties during the year • Any availability less than the following: Essential Services: 98.0% available Enhanced Services: 99.0% available

	Premier Services: 99.5% available • More than 2 major violations will force escalation procedures between the COR and the contractor. Following escalation procedures additional missed targets may be cause for termination.

Service Name	SLA 2.0: Restoration of Service
Service Description	Restoration of Service involves the implementation of procedures that ensure critical business operations resume following a disaster and that they return to normal as soon as possible. Service restoration is part of an organization's COOP plan.
Reason for Measuring	Restoration of Service is measured to ensure that systems can meet the recovery times and resume full operations within acceptable time limits based on the criticality of the application.
Time Frame	The time frame of measurement is from the time that the application is no longer available until the application is fully restored (operating in accordance with SLA defined performance criteria).
Scope	Restoration of Services applies to all of the components (hardware and software) that are required to access and run the application.
Performance Category	2.0 Restoration Time
Performance Metric	The metric used to measure compliance with restoration services is the amount of time from when services were terminated to when the end user can access and fully utilize an application.
Threshold Levels	The thresholds are as follows: Enhanced: Less than 5 days Essential: Less than 48 hours Premier: Less than 4 hours Premier with Remote High Availability: Less than 15 minutes
Formula	The amount of time from the initial disaster report until the application can be accessed and utilized to its full functionality by an end-user.
Assumptions	The contractor will notify the CTR and program manager as soon as possible after a disaster occurs. Help desk personnel should also be notified so they can inform users reporting problems with the application.
Contractor Responsibility	The Contractor must work with the Program Manager's staff to help define the recovery requirements and then to document the procedures for the Resumption of Service for the system in a Disaster Recovery Plan. The Contractor must test the Disaster Recovery Plan for the systems annually and provide a summary of the test to the CTR.

	The contractor must have accurate, timely hardware and software configuration data as well as application and system software implementation procedures.
Customer Responsibility	The Program Manager must define the level of criticality of the application being hosted and work with the Contractor to define the Disaster Recovery Requirements. The Program Manager must ensure that any government employees needed to restore an application be available in the event a disaster occurs and that they participate in the annual testing. In the event that government personnel are not able to assist in the application recovery efforts, the program manager is responsible for providing loading instructions and test scripts to ensure that the application is functioning correctly after the application is installed in the new environment.
Frequency	Disaster recovery will be tested annually. This SLA will apply when a disaster occurs.
Measurement Techniques	The Resumption of Service is measured by adding the total minutes that it takes from the time a disaster is recognized as having occurred (defined as the time that service was no longer available) to the time the system has resumed business operations (defined as services are resumed to full SLAs). The CTR will check with the help desk to determine if a trouble ticket has been opened for the applications affected by the disaster. If a trouble ticket has been opened, the CTR will use that trouble ticket as a start time for measuring the time of disaster. If a trouble ticket has not been opened, the CTR will initiate the trouble ticket for the application(s). The Service Provider will notify the CTR when the applications are ready for operation (this assumes the application was tested using the test scripts). If test scripts were not available, any time between when the application is available for testing and the time that the program management staff performs a functional test of the application will not be held against the Service Provider unless the tests fail. The trouble ticket should be closed after resumption of operations.

Reports	1. Disaster recovery test results 2. Disaster recovery plan 3. Trouble tickets
Person Responsible for Verification	The CTR will be responsible for determining a time when the application was not available due to a disaster, and when services were resumed to SLA defined standards.
Escalation Procedures	If services exceed thresholds, the CTR will be notified.
Contractual Exceptions	None
Penalties/Rewards	Minor penalty: 5% of monthly rate • Threshold values exceed agreed upon rates. Major Penalties: 25% of monthly rate • Restoring services violated thresholds by more than 20%. • 5% of monthly rate will be penalized for each day after a major penalty is assessed. The CTR and program manager have the discretion on whether to apply any penalties.

Service Name	SLA 3.0: Help Desk Service Reporting
Service Description	The help desk is the central point of contact for problem resolution. If a customer is experiencing any problems, or needs to request services, they must contact the central help desk for assistance. The help desk will either resolve the problem while they are on the phone, or they will generate trouble call tickets to assign the problem or task to the appropriate point of contact. Under the Navy/Marine Corps Intranet (NMCI), the Navy has outsourced personal computers and infrastructure to EDS. As a result any end-user problems will start with the NMCI help desk. If the problem appears to reside within the host environment, the NMCI help desk will pass the trouble ticket to the contractor's help desk.
Reason for Measuring	The help desk is the central point of contact for problem resolution. They are the direct interface to the end-user. The help desk collects metrics needed to identify problem areas, and to provide the quality assurance that is needed to ensure that customers are supported. The trouble tickets that are generated indicate problems that may extend beyond a single caller. Prompt response by the help desk may avert more problems. The help desk not only collects information on problems through the generation of trouble tickets, but they also provide an initial resolution to problems by answering questions, or guiding users through procedures. Help desk performance must be measured to ensure the end-users are receiving the support they require, trouble tickets are being accurately generated, and action is being taken to let users know the status of their trouble tickets. Trouble tickets are one way to measure availability. It is possible that a server and application are operating within established performance thresholds, but the aggregate of the various components are affecting the performance of the application. The end-user can contact the help desk to report the application's poor performance.
Time Frame	Help Desk service will be measured during support hours. The default is 24 x 7.
Scope	Under NMCI, the help desk will take the initial call, and will pass a trouble ticket to the contractor help desk if the

	problem does not involve the client piece of the application, or the client side of the infrastructure. The help desk at the host environment will take the appropriate action to resolve the problem. This SLA applies to the contractor's help desk, and does not include any actions taken by the NMCI help desk. Thresholds will be based on direct phone calls or e-mails, and trouble tickets (or similar measures) passed from the NMCI help desk. The contractor's help desk is responsible for contacting the individual submitting the trouble call if additional information is needed. The help desk is also responsible for providing feedback on efforts to fix the problem, and to provide an estimated problem resolution time. When the problem is resolved, the help desk will close out the trouble ticket. In some cases the contractor's help desk will service requests directly from the CTR, ISSM, program manger's staff, and software developers/maintainers. The vast majority of telephone calls will be for services, instead of reporting problems. Most problem calls are initiated by the end-user, and they should initially be routed through the NMCI help desk. Software exists that can monitor every incoming call to determine an average time to respond, dropped call rate, time on hold, and average length on time responding to callers. Unfortunately this software is very expensive. If the contractor already has this software, then metrics can be revised to take advantage of that monitoring capability. However since the NMCI help desk will field most calls, the cost to collect these metrics is not justified. Instead the help desk metrics in this SLA will concentrate on the response to the passed trouble tickets and the response to phone calls will be based on surveys taken from end-users.
Performance Category	3.0 Help Desk Availability
Performance Metric	This is a measurement of the availability of the help desk to respond to requests or problems. The metric used will be the probability expressed as a percentage that the help desk will answer a call, or receive and process a trouble ticket passed from the NMCI help desk.
Threshold Levels	The following are the thresholds for help desk availability:

	Essential - Premier: 99% Automatic answers to voice mail are not acceptable for contractor help desk operations.
Formula	The formula will consist of dividing all phone calls, e-mails or passed trouble tickets that the contractor's help desk has taken action on divided by the total calls, e-mail or trouble tickets sent to the contractor's help desk.
Assumptions	The NMCI help desk will be able to pass trouble tickets to the contractor's help desk. The NMCI help desk software is Remedy. The contractor's help desk must be able to interface with Remedy©, or another method of passing the trouble tickets will have to be developed and approved by the government.
Contractor Responsibility	The contractor should have a system to ensure that trouble tickets passed from the NMCI help desk are received by the contractor's help desk.
Customer Responsibility	If the end-user is experiencing problems with an application, the problem needs to be routed through the NMCI help desk. The contractor's help desk will primarily respond to trouble tickets from the NMCI help desk and phone calls requesting hosting specific services.
Frequency	Monthly
Measurement Techniques	The total trouble tickets sent from the NMCI help desk to the contractor's help desk will be gathered from the NMCI help desk software. Tickets received will be gathered from the contractor's help desk software. The measurement of phone calls answered will be gathered from interviews and spot checks by the CTR.
Reports	1. Trouble tickets from the NMCI help desk 2. Trouble tickets from the contractor's help desk
Person Responsible for Verification	The CTR will verify the contractor's help desk availability.
Performance Category	3.1 Initial Feedback
Performance Metric	This is the period of time from submission of the trouble call until the caller is notified that a trouble ticket has been filled out, and an estimated completion time is given. Feedback is generally provided in the form of an e-mail with the information that is contained on the trouble ticket. This allows the caller to verify that the information on the trouble ticket is correct, and it provides the caller with an anticipated resolution time. The feedback must also categorize the problem and provide the agreed upon

	resolution time frames.
Threshold Levels	The following are the thresholds for initial feedback: Essential - Premier: Less than 15 minutes
Formula	Time trouble ticket is completed minus the time the e-mail is sent. Measurements are in whole minutes. For example, if the trouble ticket was finished at 10:20am and the e-mail was sent at 10:29, then the time period was 9 minutes.
Assumptions	The help desk software program must have the capability to e-mail the caller the trouble ticket, or the e-mail of the end-user reporting the problem must be contained in the trouble ticket passed from the NMCI help desk.
Contractor Responsibility	When feedback is provided to the caller, a copy of the e-mail should be sent to the CTR.
Customer Responsibility	If there are problems with the trouble ticket as it was passed, or if the end-user disagrees with the categorization of the problem, the end-user needs to respond to the e-mail outlining the issues. A copy will be sent to the CTR. If the CTR disagrees with a categorization of the problem, the CTR needs to contact the contractor and resolve the issue.
Frequency	The data will be gathered over the period of 1 month.
Measurement Techniques	The CTR will utilize the feedback e-mails to determine the time periods of the feedback.
Reports	1. Trouble tickets 2. E-mails received from the contractor's help desk
Person Responsible for Verification	The CTR is responsible for verification.

Performance Category	3.2 Repeat Problems
Performance Metric	This is a measurement of the accuracy with which problems are resolved. When a trouble ticket is closed out, the problem should be investigated and corrected. Repeat problems are those problems that have been reported via a trouble ticket that have occurred again within 30 days from the close out of the trouble ticket.
Threshold Levels	The following are the thresholds for repeat problems: Essential - Premier: 05% Problems that reoccur within a 30-day window will be counted against the month in which the problem reoccurred.
Formula	Number of repeat trouble calls divided by total trouble calls. For example if 5 trouble calls had to be reworked, out of a total of 100 trouble calls, the formula would be as follows: (5/100)*100 = 05%

Assumptions	In some cases the problem will require in-depth problem analysis. Rebooting the system will not allow a root determination of the problem. The program manager and the contractor will determine when in-depth analysis should be performed. If the program manager is reluctant to perform in-depth analysis, and is comfortable with rebooting the system to solve the problem, then the CTR after receiving concurrence from both parties will not count those faults towards this SLA.
Contractor Responsibility	The contractor needs to notify the program manager when there appears to be a recurring problem that cannot be solved without in depth trouble shooting.
Customer Responsibility	When recurring problems are occurring, the program manager needs to make the determination on whether they need to conduct in-depth root cause analysis when the next fault occurs.
Frequency	Every quarter.
Measurement Techniques	The CTR will receive copies of the trouble call feedback e-mails, which can be used to determine reoccurring problems. In addition interviews with program management staff and end-users will be conducted to determine if the root cause for different problems are the same.
Reports	1. Trouble ticket feedback e-mails 2. Monitoring tools
Person Responsible for Verification	The CTR is responsible for verification.
Escalation Procedures	Issues will be brought to the attention of the CTR. The CTR can escalate the issue to the COR if it cannot be resolved at the CTR level.
Contractual Exceptions	Problems that require in-depth analysis will be excluded from the total of reworked trouble tickets. This exclusion will require concurrence from the contractor and program manager.
Penalties/Rewards	Minor penalty: 5% of monthly rate • Threshold values exceed agreed upon rates. Major Penalty: 15% monthly rate • 3.0 Help Desk Reliability Essential - Premier: Less than 95% • 3.1 Initial Feedback Essential - Premier: Less than 45 minutes • 3.2 Repeat Problems

	Essential - Premier: 10% Penalties will be levied at the discretion of the CTR.

Service Name	SLA 4.0: Problem Resolution
Service Description	Problem resolution measures of the contractor's ability to identify, respond, and correct problems or issues that affect compute services.
Reason for Measuring	Problem resolution is a portion of the mean time to repair (MTTR), which factors into overall availability. This has a direct impact on the end-user's ability to utilize the application. If the occurrence of problems remains constant, a lower MTTR will increase the operational availability of the application. Problem resolution is an important metric in measuring customer support. It measures the contractor's response time to resolving issues, as well as the skill at which they apply long-term solutions.
Time Frame	Derived by the selected hours of support. The default is 24 X 7.
Scope	This SLA measures the resolution time frames for problems reported to the contractor's help desk, or detected by monitoring software. The SLA applies to problems within the host environment. Problems are defined as a change of state in the software, hardware, or infrastructure within the host environment that adversely affects the performance of the application. Hardware or software errors that do not affect the application's performance or functionality will not be included in this SLA. The contractor will be held responsible for the resolution time on any third party hardware or software that is residing in the host environment. Problem resolution does not include problems that can be corrected by the contractor's help desk during the initial trouble report. Problems associated with the client side computer or infrastructure will be passed to the NMCI help desk, and the NMCI SLA will pertain. Problem resolution applies to the firewall, infrastructure, hardware, and software in the host environment, except the application software. Problems relating specifically to the application will be passed to the appropriate application point of contact and will not be within the scope of problem resolution.

	Priority 1 issues: Mission Critical Impact: Priority 1 issues involve critical component failure resulting in loss of application access or functionality. Examples of priority 1 issues include: faulty routers, server failure, or disk failure on a non-replicated disk. Priority 2 issues: Significant Impact: Priority 2 issues involve critical components that are degraded, or important functionality is not available. Examples include: moderate server faults where users may notice degraded system performance, failure to a replicated web server, or disk failure in a mirrored raid environment.. Priority 3 issues: Minor Impact: Priority 3 issues involve non-critical components that are inoperative, or are degraded. These are minor faults that the end-user may not noticed and cause little disruption in service. Examples of priority 3 issues include rebooting of a replicated router, restarting aborted processes, or memory short-runs. Priority 4 issues: No immediate impact. Priority 4 issues are generally non-outage situations involving requests for information. An example of priority 4 issues would be a request for the version of software on a server, or filling out a questionnaire.
Performance Category	4.0 Problem Resolution Rate
Performance Metric	The resolution rate measures the percentage of problems that are resolved within the established timeframes. Maximum response times are established to ensure all problems are resolved expeditiously.
Threshold Levels	Problem resolution rate: Priority 1 Critical: 95% Compliance with the following timeframes, no problem will exceed 12 hours. Essential - Premier: Less than 4 hours Priority 2 Major Impact: 95% Compliance with the following timeframes, no problem will exceed 24 hours. Essential: Less than 8 hours Enhanced: Less than 8 hours Premier: Less than 4 hours Priority 3 Moderate Impact: 95% Compliance with the following timeframes, no problem will exceed 4 days.

	Essential - Premier: Less than 2 days Priority 4 Minor Impact: 95% Compliance with the following timeframes, no problem will exceed 48 hours. Essential - Premier: Less than 8 hours Password Resets: 95% Compliance with the following timeframes, no problem will exceed 2 hours. Essential - Premier: Less than 30 minutes
Formula	Total number of problems resolved within the defined time frames divided by the total number of problems that have occurred. For example, 20 trouble tickets at priority 3 were received by the contractor help desk, 18 were resolved within the timeframes, 1 was resolved in 3 days, and 1 was resolved in 5 days. The formula would be 18/20 = .90. 90 percent is not in compliance, nor is the 1 trouble ticket that took 5 days to resolve.
Assumptions	The contractor's monitoring software should detect the vast majority of the problems that will affect an application's performance. The start of the problem resolutions begins when the monitoring software detects events that affect the application's performance. Another way of reporting a problem is through trouble tickets. Under NMCI the end-user will notify the NMCI help desk if there are problems with the application. If the NMCI help desk believe that the problem originates at the host environment, they will pass the trouble ticket to the contractor's help desk. The time that the contractor's help desk receives the trouble ticket from the NMCI help desk is when the time starts for problem resolution within the contractor's host environment. The contractor's help desk will categorize the problem and assign responsibilities for resolution appropriately. The contractor will be able to accept trouble tickets generated from the NMCI help desk. The contractor does not have to have the same software as NMCI, but they must have a process for receiving and responding to trouble tickets generated by the NMCI help desk. When the contractor's help desk provides feedback on a problem, they must provide a categorization of the

	problem, and the agreed upon timeframes for resolution. If the end-user does not agree with the categorization of the problem, the issue can be escalated to the CTR for resolution.
Contractor Responsibility	The contractor must have a process in place to monitor and document problems in the host environment. Documenting problems identified by the monitoring software is essential in trend analysis and long-term problem resolution. The contractor must also have a system in place to accurately categorize problems into their respective category. The contractor must have procedures in place to communicate responses and resolutions back to the NMCI help desk. In addition the contractor must provide feedback to the end-user detailing estimated resolution timeframes, based on problem severity
Customer Responsibility	The CTR must review the trouble tickets and monitoring logs to ensure that the appropriate categorization was assigned to the trouble ticket. Navy personnel or their associated contractors will assist in problem resolution with issues that may point to the application software as the cause of the problem.
Frequency	Monthly
Measurement Techniques	Response times are based on the hours of support and are calculated by subtracting the time the trouble ticket was received by the contractor's help desk to the time the trouble ticket was closed out, indicating that the problem was successfully resolved. Response times associated with problems identified by monitoring tools will start when resource thresholds are violated, or the tools indicate that application performance is degraded. Example (1) Hours of Support 24 X 7 A Priority 2 problem was reported to the NMCI help desk. NMCI staff determined that the problem was at the host environment. They passed the trouble ticket to the contractor's help desk. The contractor received the trouble ticket from NMCI at 16:55. The contractor responds at 17:05 Response time = 10 minutes The response time is calculated by subtracting the time the trouble ticket was received from NMCI from the time the contractor responded to the problem.

	17:05 – 16:55 = 10 minutes Example (2): Hours of Support = 5 X 9 (08:00 – 17:00) Priority 2 problem reported in a monitoring toolat 16:55. Contractor Responds at 08:05 the next day. Response time is 10 minutes The response time is calculated by subtracting the time of threshold violation 16:55 from the end of the hours of support for that day 17:00, and then adding the difference between the start of the hours of support for the following day and the time the response was made. (17:00 – 16:55) + (08:05 - 08:00) = 5 + 5 = or 10 minutes. The CTR will review monitoring logs and trouble tickets received from the NMCI help desk, as well as those that may have been called directly into the contractor's help desk to determine resolution timeframes. In some cases developers will notice problems with the servers, and they should interface directly with the contractor help desk.
Reports	1. NMCI Trouble tickets 2. Contractor's trouble tickets 3. Monitoring tool reports
Person Responsible for Verification	The CTR is responsible for verification.
Escalation Procedures	The CTR must be contacted if the maximum time frames for problem resolution are exceeded. If there are disputes concerning the categorization of problems, the CTR will resolve the issue. It is important that all parties understand how to categorize the severity of the problems before application support begins.
Contractual Exceptions	Response times are only applicable during support hours.
Penalties/Rewards	Minor penalty: 5% of monthly rate • Threshold values exceed agreed upon rates. Major penalty: 20% monthly rate • Threshold values fall below 85% compliance for any of the timeframes. • Problem resolution is more than twice the agreed upon maximum response time.

Service Name	SLA 5.0: Request Management
Service Description	Request management measures the contractor's ability to respond to service requests from the government. The contractor must have a process in place to receive requests, perform requirements review to ensure they understand the request, execute the request, track execution status, , and report request completion.
Reason for Measuring	The government expects quality service. One type of service is request management, which measures the speed with which a contractor reacts to and completes a service request. Consistent time frames for implementing service requests, such as complex configuration changes are needed to accurately forecast completion times. Request metrics can be used in project scheduling, budgeting, and planning.
Time Frame	Derived by the selected hours of support. The default is 24 X 7.
Scope	Request services apply to requests that effect host environment hardware and software, and do not apply to application software. Examples of request services include: Platform design services, hardware configuration changes, large-scale software maintenance (e.g., upgrading to a new operating system), or software maintenance that involves coordination between client and server software releases (such as changing to a new version of a DBMS). Request services do not cover requests associated with problem resolution nor does it cover requests for normal software maintenance. Those areas are covered under separate SLAs. Level 1 High Application Impact: Examples of level 1 requests are changes that have a significant impact on the majority of end-users, , are difficult to reverse once they are applied, are highly complex such as designing platform solutions, or require a great deal of coordination. Level 2 Moderate Application Impact: Level 2 requests affect the application, but not the end-users. Examples of level 2 requests are modifications to peripheral hardware, adding additional agents to monitor resources, adding

	additional server resources, or installing shared services. Level 3 Minor Application Impact: Level 3 changes have little, if any, impact on the application itself. Examples are modifications to the infrastructure such as modifying the access control list in the firewall, requests for facility access, adding user identification/passwords for access to the server, and routine requests that do not fall anywhere else.
Performance Category	5.0 Response Time
Performance Metric	The metric measures the compliance with adhering to the time frames established for responding to requests.
Threshold Levels	Level 1 Major Application Impact: Essential - Premier: 15 Days to develop and propose a project plan. Resolution time frames will be negotiated between the government and the contractor. Level 2 Moderate Application Impact: Essential - Premier: 5 Days to develop implementation plan, 10 Days to complete request. Level 3 Minor Application Impact: Essential - Premier: 2 Days to complete request.
Formula	Calculate the time that the trouble ticket was initiated until the trouble ticket was closed out, indicating that the request was performed to the customer's satisfaction.
Assumptions	Funding for any requests that are not covered within the scope of the contract will be negotiated separately. The timeframes in this SLA will not be impacted by the time it takes to successfully negotiate for additional services. This includes the time it takes the contractor to develop an estimate of the costs associated with executing the request. The government and the contractor agree on the level of the request and the Change Review Board approves any proposed configuration changes. Level 1 request completion times will have to be negotiated separately. Estimated completion times will have to consider complexity, operational schedules, and coordination concerns. Both the government and the contractor will agree to the estimated project completion times.

Contractor Responsibility	The contractor must provide the documented policies and procedures for submitting changes and requests. The procedures will include the use of the contractor's help desk to record the initial request for service on a trouble ticket. Trouble tickets will be used to measure the time the request was submitted until the request was completed. The contractor must also provide a coordinator to manage the requests.
Customer Responsibility	The government will submit requests in compliance with the documented policies and procedures. The CTR will determine the request level. If the distinction is not clear, the CTR, contractor and program manager can negotiate a response time that is acceptable to all parties.
Frequency	Monthly
Measurement Techniques	Times are calculated by subtracting the time the trouble call is submitted until a project plan is delivered, and/or the request is completed. The total number of requests will be categorized into those that met the threshold levels and those that did not. The numbers will then be utilized in the formula to determine compliance.
Reports	1. Trouble tickets
Person Responsible for Verification	The CTR is responsible for verification.
Performance Category	5.1 Project Completion
Performance Metric	The metric used is a percentage of time that the actual project completion date deviated from the estimate in the project plan.
Threshold Levels	The thresholds apply to the timeframes established by SLA 5.0, or to the timeframes presented in the approved project plan. The following thresholds represent an acceptable percentage deviation from the promised completion date: Essential: 15 percent Enhanced: 15 percent Premier: 10 percent
Formula	The difference between the actual time to complete the request (AT) minus the estimated time to complete the request as outlined in the project plan (ET) divided by the estimated time. Formula = (AT – ET)/ET * 100

	Actual time = 17 days Estimated time = 14 days Formula = (17-14)/14 * 100 = 21.43 percent
Assumptions	The government and the contractor agree on the project completion estimates before the contractor agrees to perform the request. Additional requirement or scheduling changes by the government will require a renegotiation of the estimated completion times. Level 1 tasks that can be performed in less than 10 days will default to level 2, and the thresholds for level 2 will apply. The time of request completion will be entered on the trouble ticket and the job will be closed out.
Contractor Responsibility	The contractor will provide an estimate of the time it will take to complete the request. The estimate will be part of the project or implementation plan.
Customer Responsibility	Review the estimated completion time to determine if the time frames meet operational commitments. Agree on time frames for completion before any work is actually performed. Allow the contractor adequate time to properly scope and research the request. What may appear to be a simple request may in fact be very complex.
Frequency	This SLA will apply to every request on a case-by-case basis. The CTR will apply any penalties at the end of the month in which thresholds were violated.
Measurement Techniques	The actual completion times for a level 1 request (taken from the trouble ticket) will be compared to the project completion estimate in the project plan. If the time actually completed exceeds the estimate, then the percentage of time difference needs to be computed.
Reports	1. Trouble tickets 2. Implementation plans: As they are developed
Person Responsible for Verification	The CTR will be responsible for verification.
Escalation Procedures	Any disputes will be resolved by the CTR. If there are still conflicts, the COR will make the final determination.

Contractual Exceptions	None
Penalties/Rewards	Minor penalty: 5% monthly rate • Threshold values exceed agreed upon rates. Major penalty: 15% monthly rate. • 5.0 Response Time Level 1 through level 3: compliance rate less than 85%. • 5.1 Project Completion Level 1: Project completion time exceeds 25% for Essential and Enhanced, and 20% for premium. • 5.1 Project Completion Level 2 and level 3: Time to complete the request exceeds 25% of the threshold.

Service Name	SLA 6.0: Security Management
Service Description	Security Management Services are those services required to protect the confidentiality, integrity and availability of the compute environment. The services include vulnerability assessments, intrusion detection, virus protection and compliance with DoD, and DoN policies and procedures.
Reason for Measuring	The Internet is an inherently untrustworthy medium. Any system that has connectivity to the Internet must have defensive systems, policies, and procedures in place to protect against attack. Many applications in the government contain information that is business sensitive. The sensitive but unclassified classification assigned to that information requires that the government take aggressive steps to ensure the confidentiality and integrity of the information. Information warfare or cyber-terrorism seeks to exploit security vulnerabilities to gather information, insert erroneous information, destroy information, and disable systems. A successful attack against a system or application can result in compromised information and hours or days of down time, depending upon the severity of the attack. Determining the extent of the damage can take days or weeks. An attacker may have penetrated the system months before; so corrupted files would be incorporated into the backup tapes. Without strong security measures it can be very difficult to determine when an attack occurred, and the extent of the damage.
Time Frame	Derived by the selected hours of support. The default is 24 X 7. Security monitoring is 24 X 7 regardless of the selected hours of support.
Scope	Security management includes the firewall, network and server hardware and software within the host environment, and does not apply to application software.
Performance Category	6.0 DoD Information Technology Security Certification and Accreditation Process (DITSCAP) Certification
Performance Metric	The DITSCAP documentation outlined in DoD Instruction 5200.40 states that the environment and all applications residing in that environment must be certified. This metric measures compliancy with the DITSCAP program. The metric is a percentage expressed as the number of applications certified in accordance with the DITSCAP program divided by the total number of applications in the

	host environment.
Threshold Levels	The DITSCAP documentation includes a security risk assessment of the host environment (firewall, network, servers, and all supporting software) and each of the applications that reside in that environment. The thresholds are split between the host environment assessment and the individual application's risk assessments. The following thresholds apply to the certification of the host environment: Enhanced – Premier: 100 percent The following thresholds apply to the certification of applications within the host environment: Enhanced – Premier: 95 percent
Formula	The number of applications certified in accordance with the DITSCAP regulations divided by the total number of applications in the host environment.
Assumptions	The information in the DITSCAP documentation will be classified in accordance with the appropriate classification guide. The DITSCAP program refers to systems and not individual applications. However, the intent of the program is to gather enough information on the application to accurately determine the application's security risk. The DITSCAP documentation for the host environment will consist of the assessment of the security risks associated with the environment, and the appropriate documentation assessing the risk for each application. The contractor is responsible for the host environment assessment, and the government is responsible for the application specific documentation. At a minimum, the government activity will provide a type or system accreditation document approved by the developmental Designated Approving Authority (DAA) to be included in the contractor's host environment accreditation document. See NIST Special Pub 800-37, Guidelines for Security Certification and Accreditation of Federal Information Technology Systems for definitions.

	The government developmental (DAA) will evaluate the DITSCAP documentation, review the security risks, and determine if the system or application will be hosted in the contractor's host environment.
Contractor Responsibility	The contractor if responsible for certifying the host environment, as well as obtaining documentation from the government identifying the risks associated with the applications to be hosted in the host environment. The will be present the DITSCAP documentation to the appropriate government developmental DAA for review.
Customer Responsibility	Provide the contractor with the type or system accreditation documentation identifying security risks associated with the application. If a System Security Authorization Agreement (SSAA) already exists, provide the document to the contractor for incorporation into the contractor's accreditation documentation. If the customer needs assistance in documenting the appropriate risk information, the contractor can perform that function, however that task will be negotiated separately.
Frequency	This review will be conducted on a quarterly basis.
Measurement Techniques	The software configuration documentation will contain an inventory of all software in the host environment. The Information System Security Manager (ISSM) will spot check the configuration document against the SSAA to ensure that the proper information has been collected on the application. The ISSM will also have a listing of all applications that are hosted in the contractor's environment. Every application should have the appropriate DITSCAP documentation. The ISSM will check the periodicity of the host environment SSAA to ensure that it is renewed every three years.
Reports	1. A listing of all applications hosted with the contractor 2. The contractor's software configuration database. 3. The contractor's DITSCAP documentation that will include the host environment documentation as well as the documentation for every application.
Person Responsible for Verification	The appropriate government ISSM.

Performance Category	6.1 Adherence to Security Policies and Procedures
Performance Metric	The metric applied to security policies is based on spot checks performed by the government to validate that the contractor is abiding by DoD, DoN and contractor mandated security policies and procedures. The metric will be expressed as a percentage of spot checks showing adherence to policies divided by the total number of spot checks.
Threshold Levels	This performance category will evaluate how well the daily operations at the host environment abide by mandated security policies and procedures. Areas that will be evaluated include ensuring security changes can be traced back to approved change requests, users have the appropriate permission and access levels, passwords are the appropriate length, personnel with root access match the personnel approved to have root access, and physical security. DoD and DoN security policy states that successful intrusions must be reported. The incident report will be used as one of the spot checks for the quarter. If it is determined that the intrusion was a result of a failure to execute security procedures, then that spot check will count as a failed spot check. This review is separate from red team vulnerability assessments. The following thresholds apply to adherence to security policies and procedures: Enhanced – Premier: 95 percent
Formula	The number of spot checks indicating adherence with the mandated security policies and procedures divided by the total number of spot checks that were conducted.
Assumptions	The government will provide audit results to the contractor for comment. The contractor will take action to correct noted deficiencies. The audit results will be classified in accordance with the appropriate classification guide.
Contractor Responsibility	The government representative will have full access to all documentation, hardware, and software necessary to conduct the spot checks. The government expects full cooperation from the contractor.
Customer Responsibility	The government will provide the contractor with a

	checklist of the possible spot checks that will be performed. If discrepancies are discovered, the government will provide any necessary instructions or documentation to assist the contractor in correcting the problem. The appropriate ISSM will forward any modifications to the checklist, or any new DoD or DoN security guidance to the contractor.
Frequency	Quarterly
Measurement Techniques	The government representative will use an extensive checklist and personal knowledge to conduct the spot checks.
Reports	1. The security checklist. 2. The appropriate logs and reports to validate security procedures and policies are being adhered to 3. Configuration data to ensure security patches were installed.
Person Responsible for Verification	The appropriate government ISSM.
Performance Category	6.2 Access Revocation
Performance Metric	The metric to measure this category is the amount of time taken to remove an individual's access rights and privileges to the server.
Threshold Levels	As personnel rotate jobs, retire, or are terminated, their ability to access and/or authenticate to a server (password, PKI certificate) must be removed. This prevents hostile activity from a disgruntled worker, and it ensures that only authorized personnel have access to the server. Revoking access rights ensures that the authorized personnel are not held accountable for actions that may have been accomplished by someone no longer working with the server. This threshold applies to government personnel as well as the contractor's employees. The ISSM will notify the contractor when access rights for government employees need to be removed. Notification will be initiated through a trouble call to the server farm help desk. If contractor employees are terminated, transfer to another position that does not necessitate access to an application's server, or retire the ISSM will be notified

	within 8 working hours. The following thresholds apply to removing an individual's access rights: Enhanced – Premier: Less than 8 hours
Formula	For revocation of a government employees access rights, the time will be measured from the issuance of the trouble ticket to the completion time on the trouble ticket. If the revocation concerned a contractor employee, the time will be measured from the time the employee was removed from the project (as reported to the ISSM) until the time the employee's rights were removed. Log entries will detail the time the employee's rights were removed.
Assumptions	If a contractor employee is transferred to another position that does not need access to a server, the contractor will revoke that individual's access. The contractor will have to determine whether an internal employee needs access rights. In some cases, the contractor may want multiple employees to have access rights for redundancy purposes.
Contractor Responsibility	The contractor must notify the appropriate ISSM of contractor personnel terminated, retiring, or transferred off of the project. Notification must occur within 8 working hours after the individual has been terminated or reassigned.
Customer Responsibility	The customer is responsible for notifying the contractor of personnel that no longer need access to the server. Notification will be through a trouble ticket. The ISSM will notify the appropriate government personnel of contractor employee terminations or reassignments.
Frequency	Monthly
Measurement Techniques	The ISSM will use the trouble tickets, notification received from the contractor, and server logs to compute the formula.
Reports	1. Database of users and corresponding access rights. 2. Trouble tickets 3. Server logs
Person Responsible for Verification	The appropriate government ISSM.
Performance Category	6.3 Red Team Vulnerability Assessment
Performance Metric	The red team is a government security team that will evaluate the host environment for vulnerabilities. The metric used will be the success rate at preventing an attacker from affecting the integrity, confidentiality, or

	availability of data or systems hosted in the contractor's environment. The metric will be a percentage representing the amount of unsuccessful attempts to breach security in the area being assessed divided by the total attempts to breach security in the area assessed (for example, blocking denial of service attacks).
Threshold Levels	The red teams will test all aspects of the host environment security. They will evaluate a number of areas including, but not limited to: physical security, personnel security, firewall compliance, system penetration, planting (e.g., Trojan horse), data integrity, denial of service, virus protection, media security, communication monitoring, communication tampering, administrative security procedures, authorization violation, and authentication. Successful red team attacks against components that are in full compliance with DoD/DoN guidance and industry standards will not count against threshold figures. Threshold levels are as follows: Enhanced – Premier: 99.00 percent
Formula	The number of unsuccessful attacks divided by the number of total attacks. An attack is defined as an attempt to exploit a vulnerability by utilizing one form of attack. For example using a war dialer to determine the phone numbers of the modem bank constitutes one attack, even if 10,000 phone numbers were dialed. Denial of service attacks against one port constitutes one attack even if numerous messages were sent to that port.
Assumptions	The first red team assessment will be used as a training mechanism, and will incur no penalties for identified vulnerabilities. The red team will provide a brief to the contractor's management to explain the purpose of the assessment and to get their authorization to conduct the test. The red team will also provide a debrief explaining the results of the assessment. Government personnel will also be invited to the briefs. The results of the red team assessment will be classified in accordance with the appropriate classification guide.

	The red team assessment will have minimal impact on the applications residing in the host environment. If SLAs are affected as a result of the red team assessment, the contractor will not be penalized.
Contractor Responsibility	The contractor will provide full cooperation with the red team, including granting full access to the host environment (it is assumed that they will be escorted).
Customer Responsibility	The customer will provide the contractor with the vulnerability assessment results so appropriate action can be taken to correct or reduce the vulnerabilities identified.
Frequency	If a host environment has not received a red team assessment within 1 year, then the assessment should be done before the application becomes operational. Otherwise the periodicity is annual.
Measurement Techniques	The red team results will contain the information to apply to the formula. The red team will determine if an attack was successful.
Reports	1. Red Team vulnerability assessment
Person Responsible for Verification	The red team will perform the assessment, and the ISSM will verify the results against thresholds. If the ISSM does not have the appropriate security clearance to view the results of the assessment, then the verification will be conducted by a member of the Chief Information Officer's (CIO) staff with the appropriate clearance.
Performance Category	6.4 Correction of Red Team Identified Vulnerabilities
Performance Metric	The metric is the number of days to correct a deficiency or vulnerability identified in the red team attack.
Threshold Levels	The time to correct deficiencies should be prioritized by the criticality of the vulnerability, and the risk it presents to the application. Critical Vulnerability: The application is at risk from an attack that is commonly utilized (hackers have used the vulnerability to attack organizations more than 30 times). This categorization is subjective and will depend upon the red teams assessment of the vulnerability and the criticality of the application. The red team will make this determination. Moderate Risk: The vulnerability has been exploited in the past, but its risk is not high. The application would be affected, but not for any significant time (over 1 day). A denial of service attack would be an example of this type of risk. This is also a subjective assessment and the red team will make this determination.

	Non-critical Vulnerability: All other vulnerabilities identified by the red team. The time thresholds are as follows: Critical Vulnerability: Essential – Premier: 5 days Moderate Risk: Essential – Premier: 14 days Non-critical Vulnerability: Essential – Premier: 21 days Successful attacks against an application will have a direct impact on availability computations.
Formula	The time, expressed in days, from the red team debrief until the vulnerabilities are corrected, verified, and reported to the ISSM.
Assumptions	The red team will debrief the contractor on all identified security vulnerabilities. The red team will be available to answer questions from the contractor after the debrief.
Contractor Responsibility	Trouble tickets should be initiated to record actions necessary to correct vulnerabilities. The description on the trouble tickets does not have to detail specific vulnerabilities (e.g., tasks necessary to correct discrepancy #5). The contractor will correct the vulnerabilities and notify the ISSM when each is corrected.
Customer Responsibility	The ISSM will verify when the vulnerability has been corrected. The ISSM should be able to accomplish verification by physical inspection, working with the red team to replicate the attack, discussing the issue with the contractor staff, or talking to the red team personnel and describing the corrective action.
Frequency	Annually
Measurement Techniques	The time is measured from the day after the red team debrief until the CTR has verified that the vulnerability has been corrected. The trouble tickets will be used to measure completion times.
Reports	1. Red Team vulnerability assessment 2. Appropriate logs and reports necessary to verify that vulnerabilities were corrected. 3. Trouble tickets
Person Responsible for Verification	The appropriate government ISSM.

Performance Category	6.5 Incidence Reporting
Performance Metric	The period of time from detection of a security breach to the report of that incident. It is the contractor's responsibility to provide security for the application. The purpose of reporting an incident to the Fleet Information Warfare Center is to capture information and generate statistics concerning cyber-attacks on government assets and data. The information also helps to determine the extent of the attack or the resultant damage (e.g., worm attacks).
Threshold Levels	Incident definitions and categories are outlined in the CJCSM 6510.01 of 15 March 2002. The corresponding timeframes and method of reporting are outlined in table B-10 of that same document. Reports will be made to the Fleet Information Warfare Center (FIWC), and the ISSM assigned to the activity of the application supported. The CJCSM 6510.01 states the information required for the report. The ISSM is notified within 4 hours of the incident: Essential – Premier: 100%
Formula	The time expressed in minutes from the initial detection until a report is properly filed (in accordance with CJCSM 6510.01).
Assumptions	Taking action to mitigate the impact of an incident takes precedence over reporting criteria.
Contractor Responsibility	Upon detection of an incident, the contractor will make an initial report within the timelines outlined in CJCSM 6510.01. If all information is not available within the timeframes, submit a partial report, and follow up later when all of the information is known. The contractor will notify the appropriate ISSM of the incident as soon as possible (no more than 4 hours after the incident).
Customer Responsibility	The customer will provide the incident reporting documentation, and all points of contact for incident reporting. The customer will provide the contractor training on how to respond to incidents and fill out the appropriate forms. The customer will provide the contractor with recall numbers to notify the appropriate government personnel in the case of an incident.
Frequency	As an incident occurs. Each incident will be measured individually.
Measurement Techniques	Security logs from the firewall, network and servers will be reviewed to determine when an incident has occurred. The initial report will also indicate the time of discovery.

	If the security logs do not indicate an incident, the time on the report can be used. The ISSM will compare the time the contractor provided notice, to the time of incident discovery to determine the threshold for notifying the ISSM.
Reports	1. The appropriate security logs 2. Reports from monitoring tools 3. Reports from FIWC 4. The incident report generated by the contractor
Person Responsible for Verification	The appropriate government ISSM.
Performance Category	6.6 IAVA, NAVCIRT, and INFOCON Response
Performance Metric	The time measured in hours from when the government notifies the contractor of an Information Assurance Vulnerability Alert (IAVA), Naval Computer Incident Response Team (NAVCIRT) advisory or Information Condition (INFOCON) action, and when the action has been completed.
Threshold Levels	IAVAs, NAVCIRTs and INFOCON advisories are issued to prevent security incidents from occurring. These advisories identify newly discovered or recently exploited vulnerabilities and outline action to correct or mitigate those vulnerabilities. Each advisory gives a time frame for complying with and reporting the actions outlined in the advisory. In the case of INFOCON alerts, compliance may be required within the hour, but these are rare occurrences. The timeframes for complying and reporting compliance will determine the threshold timeframes. Reports will be made through the activity ISSM.
Formula	The time period from when the advisory was reported as a trouble call and the time that compliance was reported to the ISSM.
Assumptions	If any of the actions mandated by an advisory adversely affects the operation of the host environment, (e.g., interferes with monitoring agents, system settings, IDS agents) the ISSM will be notified, and a resolution will be determined.
Contractor Responsibility	The contractor is safeguarding government data. As such adherence to IAVAs, NAVCIRTS and INFOCON is required. The contractor will notify the appropriate ISSM when the actions outlined in the advisories have been completed.

Customer Responsibility	The ISSM will initiate a trouble call to the server help desk notifying the contractor of receipt of an IAVAs, NAVCIRTS and INFOCON. The ISSM will then deliver the alert to the contractor (fax, e-mail) as soon as they are received.
Frequency	Each advisory will be tracked individually.
Measurement Techniques	The ISSM will initiate a trouble call informing the contractor that they need to take action on an advisory. The ISSM will e-mail the advisory (a confirmation of receipt is required), or fax it to the contractor (a follow up phone call confirming receipt is required). The advisory will contain the time frame for compliance. That time period sets the threshold. The time from when the trouble ticket was submitted until the contractor reports compliance will be measured against the time requirement in the advisory to determine compliance.
Reports	1. IAVA, NAVCIRT or INFOCON messages 2. Trouble tickets
Person Responsible for Verification	The appropriate government ISSM.
Escalation Procedures	The activity DAA and associated ISSMs will be notified of vulnerability results. The CTR will be notified if any thresholds are violated. Any disputes will be resolved by the CTR. If there are still conflicts, the COR will make the final determination.
Contractual Exceptions	The initial red team attack will evaluate vulnerabilities and adherence to DoD and DoN policies and guidance. The results from the first vulnerability assessment will not count against this SLA. The first assessment will not only identify areas that need improvement, but will also clarify policy and procedural interpretation.
Penalties/Rewards	Minor penalty: 5% monthly rate • Any threshold values were exceeded. Major penalty: 15% monthly rate. • More than 4 minor penalties during the year. • 6.3 Success rate against red team less than 95% • 6.4 Correction of security vulnerabilities in the red team assessment or in an advisory exceeds 20% of thresholds. If time periods exceed 20% of threshold, there will be a 5% monthly rate penalty for every week until compliance.

Service Name	7.0 Software Maintenance
Service Description	Software maintenance involves installing new files, updates, or patches to the infrastructure, DBMS, and system software. For the purposes of this service level agreement, the terms patches, upgrades, and modifications are all considered maintenance actions, and the terms will mean the same. This SLA is concerned with the time it takes to realize that an upgrade to software in the host environment has been released until it is tested and finally installed in the production environment. This SLA does not cover the development of the maintenance software, nor does it cover the quality of the maintenance software. In most cases the software upgrade is from a third party vendor, and the quality of the software upgrade is a risk that the contractor must incur and manage. Software maintenance also has to be performed on the application, and its associated software. If the maintenance action requires root access to install the changes, then assistance will be required from the contractor, as only the contractor has full root control.
Reason for Measuring	Upgrades are generally released to correct problems with the software (bugs), update software to prevent new attacks, or to add/enhance functionality. The security of the application is dependent upon the speed at which the contractor installs security related updates. As a result it is important to place time frames on the contractor to ensure that security related patches and updates are installed as soon as possible. The contractor controls root access to the server. Application maintenance action requiring root access must be coordinated with the contractor. The threshold time frames are designed to give the contractor sufficient time to have staff available to assist with the installation of the application update. The government's maintenance personnel also have consistent response time frames that they can use to schedule their maintenance.
Time Frame	Derived by the selected hours of support. The default is 24 X 7.
Scope	Software maintenance covers all system, DBMS and

	infrastructure software. The software maintenance is only concerned with the software that resides in the host environment, and is not concerned with the client side of the software. Software maintenance concerns patches and upgrades to system and infrastructure software. The upgrades are not new releases of the software, but are supplements to existing installed versions. Upgrades to an existing version, (version 2.0) of application X, would be covered by this service level agreement, whereas installing a new version, (version3.0) would fall under the service level agreement for software refresh. Maintenance actions initiated by the government will not be constrained by this SLA. However, government initiated down time will not count against availability or contractor initiated maintenance time. Maintenance action to the application that does not require root access is not covered under this SLA. Tuning operating system software is not covered under this SLA. Tuning is considered a routine operation necessary to host an application.
Performance Category	7.0 Installation Time Frames
Performance Metric	The metric is the amount of time from release of a patch or update, until it is tested and installed.
Threshold Levels	System, DBMS, and infrastructure software installation priorities are as follows: Priority 1: Critical Security Related Patches. An example would be alerts covered under an IAVA or NAVCIRT. However, government generated alerts are covered under another SLA. This SLA is concerned with third party vendors, or the contractor, releasing patches in response to newly identified vulnerabilities. Priority 2: Routine Security Patches. Examples are virus or IDS signature updates. Priority 3: Upgrades correcting known errors: Examples are upgrades correcting functional problems, such as interfacing with new drivers.

	Priority 4: Routine upgrades or patches: Examples are upgrades adding new functionality. Thresholds are as follows: Priority 1 Maintenance Action: Essential – Priority: Within 8 hours from release from third party vendor. Priority 2 Maintenance Action: Essential – Priority: Submit to test lab within 1 day after release. Install within 3 days of release. Priority 3 Maintenance Action: Essential – Priority: Submit to test lab within 1 week after release. Submit the maintenance action to the configuration review board (CRB) at the first opportunity. Install within 1 week from CRB approval. Priority 4 Maintenance Action: Essential – Priority: Submit to test lab within 2 weeks of release. Submit the maintenance action to the configuration review board (CRB) at the first opportunity. Install within 1 week from CRB approval.
Formula	The time from the release of the patch or update to the time it is tested and installed.
Assumptions	Government personnel will notify the contractor of any priority 1 maintenance actions initiated from the government. Priority alerts from commercial sources will be the responsibility of the contractor. It is assumed that the contractor will subscribe to security alert services. If a third party's security patch is included in an IAVA, or NAVCIRT, the timeframes for installation will default to the government alert instead of this SLA. Due to the short timeframes involved with installing priority 1 maintenance actions, the CRB will be notified after the installation has been completed. Notification will be made through the government ISSM. Priority 2 maintenance actions are considered routine and part of daily business, and do not require the approval of the CRB. All maintenance actions must be properly documented.

	All maintenance actions will be annotated on the weekly schedule maintenance plan. Priority 3 and 4 maintenance actions will be performed during the maintenance window.
Contractor Responsibility	The contractor will develop procedures to ensure that the time frames are met. The contractor must annotate the release date of a patch or upgrade on the scheduled maintenance plan. All priority 3 and 4 maintenance actions must be presented and approved by the change review board. The contractor will notify the ISSM after any priority 1 patches are installed. Notification will be no later than the day following the installation.
Customer Responsibility	Notify the contractor of any government issued security alerts.
Frequency	Monthly
Measurement Techniques	The ISSM can check compliance with priority 1 maintenance actions by reviewing the trouble tickets and monitoring logs, and comparing those entries to the date the vendor released the update. The ISSM can check Internet history logs to determine if the contractor is downloading security patches on a daily basis. Software configuration documentation will list when those security patches were installed. The history logs will also ensure that the contractor is checking vendor's web sites, or monitoring security bulletins on a daily basis for new software patches or upgrades. The ISSM can check the software release dates on the scheduled weekly maintenance report, and compare those to actual release dates by calling the central design agency (CDA). The actual software release dates can be compared to the CRB notes to ensure that the maintenance action was presented to the CRB at the first opportunity. The CRB notes will contain approved maintenance action. The software configuration documentation will contain the date the software update was installed. The ISSM can check the dates to ensure the maintenance action was

	performed within 1 week of CRB approval.
Reports	1. Scheduled Maintenance Report 2. Server Logs 3. CRB Minutes 4. Software Configuration Documentation 5. Internet History Logs
Person Responsible for Verification	The appropriate ISSM
Performance Category	7.1 Root Access Assistance
Performance Metric	Only the contractor has root access to the operating system. As such, application developers needing access to files requiring root authority will have to coordinate with the contractor for access. This metric measures the time from the request for root access assistance until the application upgrade installation begins. This SLA affects application problem resolution because in some cases root access will be needed to restore corrupted or missing files.
Threshold Levels	Installation of application upgrades requiring root access is broken into three levels. Level 1: Installing Critical Application Upgrades: Examples include repairing security vulnerabilities, or significant functional errors. Level 2: Installing Serious Application Upgrades: Examples include repairing degraded functionality or performance. Level 3: Installing Routine Application Upgrades: Examples include adding new functionality. Thresholds are as follows: Level 1: Critical Upgrades Essential – Premier: Grant root access 4 hours after notification. Level 2: Serious Upgrades Essential – Enhanced: Grant root access 8 hours after notification Premier: Grant root access 4 hours after notification. Level 3: Routine Upgrades Essential – Premier: Grant root access within 3 working

	days after notification.
Formula	The time of the request for root access assistance minus the time that the application upgrade installation begins.
Assumptions	The government will perform the actual application upgrade installation. The contractor is only needed to grant root access to the government personnel. The CTR will track all government initiated maintenance actions to ensure that the maintenance down time is not charged against the contractor. The software configuration documentation will include not only the time frames for application upgrade installation, but also all pertinent information about the upgrade such as a detailed description, developer, purpose of the upgrade, and patch/version number.
Contractor Responsibility	Ensure that personnel are available and trained to grant root access during scheduled support hours. After hours personnel must be accessible by phone or pager to respond to after support hour level 1 root access requests. The contractor help desk will be used to generate a trouble ticket for root access requests. The help desk will determine the appropriate level. If there are disputes concerning level 1 requests, the contractor will grant the request and file a grievance through the CTR for resolution.
Customer Responsibility	The CTR will initiate contractor assistance through a trouble call to the contractor's help desk. The government maintenance personnel must inform the CTR if the maintenance action affected the application. If the application was impacted as a result of the maintenance action, that 'down time' will not count against availability SLAs.
Frequency	Monthly
Measurement Techniques	The CTR will review the trouble ticket to determine the time between the request and the time the contractor granted root access to the server. The trouble tickets will be grouped into the three levels and the appropriate thresholds will be applied. The CTR can also review the maintenance records and configuration documentation to the times that the software was installed.
Reports	1. Trouble Tickets

	2. Software Configuration Documentation 3. Maintenance Records
Person Responsible for Verification	The CTR is responsible for verification.
Escalation Procedures	The CTR will attempt to resolve all disputes concerning the maintenance priorities or request levels. Disputes that cannot be resolved will be presented to the COR.
Contractual Exceptions	Maintenance downtime associated with application upgrades will not count against the contractor's availability or maintenance SLA thresholds.
Penalties/Rewards	Minor Penalty: No monetary penalty • Any threshold values were exceeded. Major Penalty: 25% of monthly rate. • More than 3 minor penalties in any maintenance category in one year. • Any of the priority 1-4 response thresholds for upgrades were exceeded by more than 50%. • If any of the level 1-3 response thresholds for root access were exceeded by more than 50%.

Service Name	SLA 8: Maintenance Schedules
Service Description	Maintenance in this SLA involves hardware and software maintenance. Hardware maintenance can involve changing routers, installing memory, or repartitioning drives. Software maintenance involves installing new files, updates, or patches to the infrastructure, DBMS, and system software. This service level agreement outlines the day and the times that will be used to perform maintenance that affects the application. The SLA also specifies the amount of time that the application is affected as a result of the maintenance actions throughout the month.
Reason for Measuring	Fixed maintenance windows set a level of user expectation. Users should not expect full access to an application during scheduled maintenance windows. Maintenance down time has direct business repercussions. When an application is not functioning, users cannot perform their jobs, schedules are affected, morale declines, and opportunities are lost. Specifying maintenance windows, and the total amount of maintenance down time allows an organization to take the application down time into consideration. Activities can be planned around the scheduled maintenance down time.
Time Frame	Derived by the selected hours of support. The default is 24 X 7.
Scope	Any hardware or software maintenance actions within the host environment (including the firewall) that affect the application will apply to this SLA. Maintenance to the application itself will not be covered under this SLA.
Performance Category	8.0 Maintenance Window
Performance Metric	This is the scheduled time period in which maintenance actions can occur.
Threshold Levels	The following thresholds apply: Essential: Sunday 0800-1200 Enhanced: Sunday 0800-1200 Premier: No scheduled downtime Any maintenance action performed outside of the maintenance window will count as application down time and will be used in the availability computations.

	Any deviations from the maintenance window will have to be approved by the application program manager. The CTR must be informed of any approved maintenance activity outside of the maintenance window.
Formula	None
Assumptions	Installation of security signatures on the IDS, anti-spam and anti-virus software will not require downtime.
Contractor Responsibility	All maintenance action initiated by the contractor will be performed within the maintenance window. Notify the CTR of scheduled maintenance action during the week.
Customer Responsibility	Inform users of the application that there may be difficulties in accessing the application during scheduled maintenance windows.
Frequency	Monthly
Measurement Techniques	The CTR will review the weekly maintenance schedule from the contractor. The maintenance should all be scheduled within the maintenance window. The CTR will review monitoring logs to ensure that the application was only "down" for maintenance time within the scheduled time frames. The CTR must be informed of any negotiated deviations from the maintenance window. Application down time not within the scheduled maintenance window will count against the availability SLA.
Reports	1. Maintenance schedule 2. Trouble tickets 3. Monitoring logs
Person Responsible for Verification	The CTR is responsible for verification.
Performance Category	8.1 Maintenance Hours
Performance Metric	This is the total scheduled maintenance time for the month.
Threshold Levels	The following thresholds apply: Essential: 4 hours Enhanced: 4 hours Premier: No scheduled downtime
Formula	Add the maintenance time during which the application was affected.
Assumptions	The change management board must approve all software maintenance actions, with the exception of emergency security updates. All maintenance action is tested before installation. In the case of emergency security installation, the application is

	tested after the installation. Tests will be conducted in accordance with the approved test plan. Any system or infrastructure down time outside of the scheduled maintenance time will be considered down time and will count against availability service level agreements. For example if the scheduled maintenance down time is 4 hours, and 5 hours were actually used to perform maintenance during the month, then 1 hour will be considered down time in the availability computations.
Contractor Responsibility	Notify the CTR of maintenance actions that will be scheduled during the week.
Customer Responsibility	The customer is responsible for notifying end-users if their access to the application will be affected by scheduled maintenance.
Frequency	Monthly
Measurement Techniques	The CTR will verify the scheduled maintenance down time against the system monitoring logs. The CTR will then calculate total maintenance time by adding the maintenance down time during the month.
Reports	1. Maintenance schedule 2. Monitoring logs
Person Responsible for Verification	The CTR is responsible for verification.
Escalation Procedures	The CTR will be notified of any deviations from the maintenance windows or schedules.
Contractual Exceptions	Scheduled maintenance initiated by the government will not be applied to this SLA.
Penalties/Rewards	Maintenance action outside of the schedule maintenance window, or maintenance down time exceeding thresholds will be considered down time for availability computations. Availability penalties will apply.

Service Name	9.0 Migration Services
Service Description	Migration services are those services required to move, install, and operate an application in the contractor's Application Hosting environment.
Reason for Measuring	Transition services are measured to ensure that the project is completed on time, and that the application's performance does not suffer as a result of being hosted in the contractor host environment.
Time Frame	This SLA covers the time period from contract award until the application is installed in the production environment, can be accessed by its intended end-users, and is fully operational. The completion time will be determined when the government validates that all migration requirements have been satisfied.
Scope	Migration in the context of hosting applications is the process by which an application is transferred from one platform to another. The specific tasks that need to be performed during the migration phase and the deliverables are specified in the statement of work (SOW). The scope covers all activities necessary to migrate the application to the contractor host facility, including application audits, designing activities, performing requisite testing (outlined in the migration plan), placing the application into the production environment, establishing connectivity, and operating the application at full functionality.
Performance Category	9.0 Implementation, Integration, and Test Service (IIT) Service Window.
Performance Metric	The metric establishes the amount of time to perform all actions required to migrate an application to the contractor's host environment.
Threshold Levels	The threshold levels are as follows: Essential Services: 3 months Enhanced Services: 3 months Premier Services: 3 months
Formula	The time is measured from the date the contract is awarded and concludes at documented acceptance of migration services.
Assumptions	Actions relating to estimating migration costs will not be included in the migration time. For example audits must be conducted on the application to properly scope a bid.

	The time necessary to conduct a preliminary audit will not count as migration time. Once the contract is awarded any subsequent audits will count as migration time.
Contractor Responsibility	The contractor must coordinate with the government for functional testing and access to the application. The contractor must understand and operate within the government's operational constraints.
Customer Responsibility	After the contract has been signed, the contractor must have access to the application and current hosting facilities to perform a full audit, and to package the application. The government may have to negotiate with third parties to obtain access permission.
Frequency	The frequency spans the time from contract award until the government documents acceptance of the migration action.
Measurement Techniques	The date that the government has documented acceptance of migration services is subtracted from the date the contract was awarded.
Reports	1. Hosting contract: It will determine threshold start times. 2. Migration plan: The government will document acceptance of migration services. This document will be incorporated into the migration plan for official acceptance.
Person Responsible for Verification	The CTR is responsible for verification.
Performance Category	9.1 Application Performance
Performance Metric	The metric used to test application performance will be an industry standard benchmark test. Areas measured will include areas such as input-output times, memory paging, bandwidth utilization, and processing speeds.
Threshold Levels	Threshold levels are based on a comparison of benchmark tests run in the previous host environment with identical tests run on the application in the contractor's host environment. The following thresholds apply: Essential – Premier: Identical or greater performance in all areas of the benchmark tests.
Formula	This will be a direct comparison of the benchmark tests in the two environments. The tests in the new environment should be equal to or exceed the results obtained in the previous host environment.
Assumptions	The government and contractor will determine benchmark tests to execute to test the performance of the application.

Contractor Responsibility	If the government has not determined which benchmark tests to utilize, the contractor will recommend industry standard benchmark tests to the government. Execute the benchmark tests on the application in both host environments and provide results to the CTR.
Customer Responsibility	The contractor must have full access (root) to the application and associated servers in the previous host environment in order to run the benchmark tests. The government is responsible for obtaining the cooperation of the staff in the previous host environment. The government will monitor the testing to understand any differences in how the benchmark test was applied. In some cases the differences in the tests occur as a result of configuration differences in the host environments. The government representative will ensure the results accurately measure the application's performance.
Frequency	This measurement is from the time that the contract is awarded until the government documents that all migration requirements have been met.
Measurement Techniques	The government representative will compare the application benchmark tests in both environments to ensure that the application's performance equals or is better in the contractor host environment.
Reports	1. Benchmark test results
Person Responsible for Verification	The CTR is responsible for verification. Verification in this case may require the assistance of the application developers to ensure the tests are run correctly.
Escalation Procedures	The contractor will notify the CTR if the migration cannot be accomplished within time frame thresholds. Designated government representative will approve results of the benchmark tests. COR will resolve all conflicts.
Contractual Exceptions	None
Penalties/Rewards	Minor penalty: 5% monthly rate • Any threshold values were exceeded. Major: 15% monthly rate • Migration transition times exceed 50% of the threshold. • Application benchmark tests in the new host environment do not meet or exceed the benchmark tests in prior host environment. If there are performance issues, the application will not be placed in operation until the problems are resolved.

Service Name	SLA 10 Backups
Service Description	Backups refer to the process of copying data, files, disks, or the entire application to tape. There are two general types of backups. A full backup contains all of the data in a file system. An incremental backup contains only those files that have changed since the last backup. This service level agreement will measure the accuracy of the backup, adherence to the back up schedule, accuracy of tape labeling, accuracy of tape library, and restoration timeframes.
Reason for Measuring	Computers are not 100 percent reliable, disk drives can fail, files and data can be corrupted, and disasters can destroy the entire system. If the information stored in the file system has any value, it must be backed up. Backups act as a form of redundancy, and are designed to protect the integrity of a system's data. If a disk drive crashes, the information on the backup tapes can be used to restore the system. Restoration speed, tapes accuracy, and the accuracy of the tape library affect the MTTR, which influences overall availability of the application. There may also legal requirements for the retention of financial data, audit logs, or other data required for possible investigations or audits.
Time Frame	The time frames is 24 X 7.
Scope	Backups refer to application software, system software, DBMS, database files, and system and monitoring logs hosted in the contractor's host environment. There are numerous DoD and DoN policies and directives concerning backups, such as on-site storage requirements, and protecting the security of the data on the tape. Adherence to those policies will be covered under the security SLA.
Performance Category	10.0 Backup Schedule
Performance Metric	The metric will measure the contractor's adherence to the backup schedule. The metric will be expressed as a percentage of backups performed within the schedule divided by the total number of backups that should have been performed. Adherence to the schedule is vital in protecting the data in

	the file systems. If an incident occurs where the files are destroyed, any data received, modified, or deleted from the time between the incident and the last backup is lost. This may have serious repercussions for mission critical, data intensive systems. If the schedule is not followed, the risk of loosing business essential data increases.
Threshold Levels	The normal backup schedule is where incremental backups are performed daily 6 times a week and a full backup is performed on Saturday or Sunday. Additionally a full monthly and end of year backup are performed. Once the backup tapes are created they must be stored for a period of time before they can be reused. It is possible for a file to be corrupted and not noticed for weeks or months because the file is rarely accessed. As a result, it is prudent to keep copies of the file systems for a reasonable period of time. The following is a recommended backup schedule with storage days: Daily incremental backups must be stored for 8 days Weekly full backups must be stored for 2 months Monthly full backups must be stored for 12 months Annual full backups must be stored for 5 years. The thresholds for conforming to the backup schedule are as follows: Enhanced – Premier: 99% The thresholds for conforming to the backup storage requirements are as follows: Enhanced – Premier: 99%
Formula	The number of backups performed within the backup schedule divided by the total number of scheduled backups. The number of backups stored within the storage requirements divided by the total number of stored tapes.
Assumptions	The contractor will be responsible for providing the tape media. The media can be reused, but after a period of time, the media degrades and must be replaced. The contractor is responsible for replacing the media.
Contractor Responsibility	Brief the application program manager on the backup schedules and procedures that will be used to backup the application.
Customer Responsibility	Cooperate with the contractor in developing the backup schedule and associated backup procedures for the application.

Frequency	Monthly
Measurement Techniques	The government auditor must perform spot checks to ensure the backups were conducted within the scheduled time frames, and that they are stored for the appropriate amount of time. The auditor will check the system logs and monitoring logs to determine when the backups were actually performed. The auditor will have to physically check the tape storage areas to ensure tapes are being stored for the appropriate amount of time. The tapes must be labeled with the date of the backup, so determining the storage time is simply a matter of ensuring all of the tapes for the required storage period are present. For example, when checking the daily tapes, there should be 7 days of backups available (1 day is a weekly update).
Reports	1. Monitoring logs 2. System logs 3. Backup schedule
Person Responsible for Verification	The CTR will be responsible for verification.
Performance Category	10.1 Tape Backup Accuracy
Performance Metric	This category measures the accuracy of the tape backup. If the system is not backed up correctly, then the system's data is not protected, and data critical to the organization could be lost. Tapes have a shelf life of approximately 3 years. After 3 years the files on the tape must be transferred to new medium. The accuracy of the file transfer from the old medium to the new medium will be included in this category. The measurement will be the percentage of files that were backed up correctly divided by the number of files that were spot-checked.
Threshold Levels	Backup accuracy thresholds are as follows: Essential: 99.5% Enhanced: 99.5% Premier: 99.7%
Formula	The number of files that were accurately backed up divided by the total number of files sampled.
Assumptions	Restoration should be performed on a test platform.
Contractor Responsibility	The contractor must implement backup software that verifies backed up files by reading the files after they are written to the tape. The contractor will assist the government representative with loading the tapes to

	conduct the spot checks.
Customer Responsibility	Coordinate with the contractor for performing the spot checks. Access to a test server will be required.
Frequency	Quarterly
Measurement Techniques	The proof that the files were correctly backed up is to read and/or restore the contents of the tape. A representative sample of tapes will be evaluated. Random files will be accessed to determine if they can be read. Other files will be restored. Sample files will be evaluated from each tape.
Reports	1. Tape library
Person Responsible for Verification	The CTR will be responsible for verification.
Performance Category	10.2 Tape Documentation Accuracy
Performance Metric	Tape documentation refers to the labeling on each tape, and the tape library documentation. It is essential that each tape be clearly and accurately labeled. The tape labels will have detailed information to uniquely identify their contents. Information such as date and time of the backup along with the format of the files will also be included. The tape library records at a minimum, the files stored on each uniquely numbered tape as well as the dates the files were backed up. The metric used will be a percentage of tapes accurately labeled and recorded in the tape library. If any of the files on the tape do not match the documentation of either the tape label or the tape library, then the tape documentation is considered incorrect. Tape documentation is essential in rapidly restoring files.
Threshold Levels	The following are the thresholds for backup documentation. Essential: 97% Enhanced: 97% Premier: 98%
Formula	The formula is the number of tapes accurately labeled and recorded in the tape library divided by the total number of tapes spot checked.
Assumptions	The documentation requirements in this SLA also pertain to backup media other than tapes.
Contractor Responsibility	Provide the necessary tape library documentation to perform the spot check. Assist the government

	representative with loading the tapes to conduct the spot check.
Customer Responsibility	Coordinate the spot check with the contractor. Allow enough time for the contractor to have the equipment and staff on hand to assist with the spot check.
Frequency	Quarterly
Measurement Techniques	The tapes will be loaded onto a platform for read access. The files contained in the tapes that are spot-checked will be evaluated against the tape label and the tape library.
Reports	1. Tape labels 2. Tape library
Person Responsible for Verification	The CTR will be responsible for verification.
Performance Category	10.3 Restoration Time Frames
Performance Metric	Restoration refers to the task of retrieving a file from a backup tape and installing it on a system. The first step is to determine which tape has the version of the file needed. The individual file then has to be found and copied to the system server. The backup copy of the file then replaces the missing or corrupted file on the server. This section refers specifically to restoring application related files. Restoration time for system software will be included in the overall timeframes for system availability or problem resolution. The files being restored are part of the application; as government personnel may require root access from the contractor. The performance metric is the time from the request to restore a fileuntil the file is installed and operational. The request will be placed with the contractor's help desk.
Threshold Levels	The restoration time thresholds will depend upon the severity of the problem necessitating the restore action. Priority 1 issues: Mission Critical Impact: Priority 1 issues involves loss of application access or functionality. Priority 2 issues: Significant Impact: Priority 2 issues involve degraded application functionality. Priority 3 issues: Minor Impact: Priority 3 issues involve minor faults that the end-user may not noticed and cause little disruption in service. Priority 3 issues also involve restoration of files for inspection or audit purposes.

	File restoration thresholds are as follows: Priority 1 Critical: 95% Compliance with the following time frames, no problem will exceed 12 hours. Essential: Less than 4 hours Enhanced: Less than 4 hours Premier: Less than 4 hours Priority 2 Major Impact: 95% Compliance with the following timeframes, no problem will exceed 24 hours. Essential: Less than 8 hours Enhanced: Less than 8 hours Premier: Less than 4 hours Priority 3 Moderate Impact: 95% Compliance with the following timeframes, no problem will exceed 4 days. Essential - Premier: Less than 2 days
Formula	The number of restoration procedures performed within stated thresholds divided by the total number of restoration procedures performed.
Assumptions	When a problem occurs, the NMCI help desk will field the trouble call. The trouble ticket will be passed to the contractor's help desk. If the problem points to the application itself, the government personnel will trouble shoot the application. If a file needs to be restored, the government personnel will place a trouble call to the contractor's help desk to start the restoration trouble ticket. The restoration times associated with problems with DBMS, infrastructure, or system software will count against availability calculations, and not this SLA.
Contractor Responsibility	Cooperate with the government personnel that are restoring the application files. If root access is required, that SLA will apply. The contractor's help desk will determine the priority level of the restoration request. The level of the request will be annotated on the trouble ticket. If there are disputes covering the priority of the request, grant the request and file a grievance through the CTR for resolution.
Customer Responsibility	The government will request file restoration using the contractor's help desk. The government will work with the contractor to train the help desk personnel determine the appropriate priority levels for requests.

Frequency	Monthly
Measurement Techniques	Review the trouble tickets for restoration services and determine whether any of the requests did not meet the designated time frames. Check restore times against server and monitoring logs, if designated time frames were violated; apply the formula to determine compliance with the thresholds.
Reports	1. Trouble tickets 2. Server logs 3. Monitoring logs
Person Responsible for Verification	The CTR is responsible for verification.
Escalation Procedures	The CTR will be notified of threshold violations. If there is disagreement concerning the categorization of priorities, the CTR will work with both the contractor and the CTR to resolve the issues. If the problems persist, the issue will be referred to the COR.
Contractual Exceptions	None
Penalties/Rewards	Minor penalty: 5% monthly rate • Any threshold values were exceeded. Major penalty: 20 % monthly rate • Three minor penalties within the year • 10.0 Backup schedule compliance in each service level (essential – Premier) is below 90% • 10.1 Backup Accuracy is below 95% in each service level • 10.2 Backup documentation accuracy in each service level is below 90% • 10.3 Restoration services exceed maximum response times for the priority assigned to the service.

Service Name	SLA 11 Batch Services
Service Description	Batch processing used to refer to the processing of a batch of punch cards. Today the term is used more to describe the sequential processing of data. Typically once a batch job begins, it continues until it is done or until an error occurs. The next sequential program is then run, until all programs have executed fully. Many financial programs contain batch processing, especially during reconciliation processes.
Reason for Measuring	Batch jobs require additional oversight because they must be run in sequence, and they usually must be run within specified time windows. When batch jobs are running, there is no user input into the program. As a result it is important that batch jobs are run efficiently, because users are locked from the program while the batch jobs are processing. Additionally, if any errors occur while processing a batch job, it must be run again, and any information processed must be either backed out, or over written.
Time Frame	The time frames is 24 X 7.
Scope	Batch jobs will be identified to the contractor during the migration audit. The contractor is responsible for maintaining a batch job schedule, which lists the batch job, and the time frames allotted for processing. This service level agreement refers to the batch jobs contained on the batch schedule. Maintaining a batch schedule is a systems administrator function, even though it directly supports an application, or its associated databases. As such, it is the responsibility of the contractor to run the batch jobs.
Performance Category	11.0 Batch Accuracy
Performance Metric	The batch job should execute as desired. If errors occur in the process, then the process should be run again. The contractor is responsible for monitoring batch program execution. The performance metric is a percentage of the programs executed within specifications divided by the total number of programs executed. Each sequential program is distinct. If the entire batch contains 15 sequential programs, then each program will be counted individually.
Threshold Levels	The thresholds for batch processing accuracy is as follows: Essential: N/A Enhanced: 99.5%

	Premier: 99.7%
Formula	The batch programs executed within specifications divided by the total number of programs executed.
Assumptions	The contractor must perform, or assist the government in batch program restarts. Detailed execution procedures will be developed for each batch job. If problems with the batch job persist, the contractor will notify the designated government personnel.
Contractor Responsibility	Ensure the batch job schedule is accurate, and the staff is properly trained to execute the batch programs.
Customer Responsibility	Ensure that the batch job schedule contains all of the batch jobs that pertain to an application. Provide the contractor all pertinent information to execute and monitor the batch jobs. This includes providing test scripts or a description of the expected output to ensure the program is executing to specifications.
Frequency	Monthly
Measurement Techniques	The CTR will review the batch processing monitoring reports and evaluate trouble tickets that may pertain to the batch jobs.
Reports	1. Trouble tickets 2. Monitoring logs 3. Server logs 4. Batch job schedule
Person Responsible for Verification	The CTR is responsible for verification.
Performance Category	11.1 Batch Job Completion
Performance Metric	Many batch jobs must be completed within a specific time window. The metric will be presented as the percentage of batch jobs executed successfully within the scheduled time frames. Recommended time frames are as follows: All daily, weekly and monthly batch runs must be completed by 0700 AM of the following business day. If a batch job is not completed by the deadline, the contractor and government must determine if the batch job should still be run, or if it should be terminated.
Threshold Levels	The thresholds for batch job completion are as follows: Essential: N/A Enhanced: 95% Premier: 95%
Formula	The formula will be the number of batch jobs executed within the scheduled time frames divided by the total number of batch jobs scheduled to be executed.

Assumptions	Government requests for batch job execution for jobs not listed on the schedule will not count against this SLA. The recommended time frames for batch processing will be modified to suit the needs of each application.
Contractor Responsibility	Notify the government representative if a batch job cannot be completed within the scheduled time frame.
Customer Responsibility	Work with the contractor to determine a course of action if a batch job is not processed by the deadline.
Frequency	Monthly
Measurement Techniques	Review the batch job schedule and the batch job monitoring report to determine any processing outside of the scheduled time frames. Divided the number of batch jobs completed within the time frames by the total number of scheduled batch runs.
Reports	1. Monitoring logs 2. Server logs 3. Batch job schedule
Person Responsible for Verification	The CTR is responsible for verification.
Performance Category	11.2 Batch Job Requests
Performance Metric	This category is concerned with the addition, deletion, modification, or stopping of a batch job. The batch job schedule may need to be modified for a number of reasons, including seasonal requirements, new regulations, changing business processes, new requirements, or errors were found in the program.
Threshold Levels	Response times for request to add to, delete from or modify the batch job schedule are contingent upon the impact that the batch job has to the organization's business process. Priority 1 issues: Mission Critical Impact: Priority 1 issues involves a critical impact to business processes. Priority 2 issues: Significant Impact: Priority 2 issues have a noticeable impact on business processes. Priority 3 issues: Minor Impact: Priority 3 issues are routine adjustments to the batch job schedule. Stop Action: There are instances where the batch jobs should not be run as scheduled. The government must give the contractor proper notification before the contractor can stop the batch job.

	Request response thresholds are as follows: Priority 1 Critical: 95% Compliance with the following timeframes, no request will exceed 12 hours. Essential: N/A Enhanced: Less than 4 hours Premier: Less than 4 hours Priority 2 Significant Impact: 95% Compliance with the following timeframes, no problem will exceed 24 hours. Essential: N/A Enhanced: Less than 8 hours Premier: Less than 8 hours Priority 3 Moderate Impact: 95% Compliance with the following timeframes, no problem will exceed 5 days. Essential – Premier: Less than 3 days Stop Action: Essential – Premier: The batch process will not be run if notification is given 1 hour before the scheduled run.
Formula	The number of requests that were satisfied within the time frames divided by the total number of requests.
Assumptions	Any requests to modify the batch jobs will have to be requested through the contractor's help desk.
Contractor Responsibility	Work with the program manager in determining criteria for categorizing the criticality of batch job requests.
Customer Responsibility	Give the contractor as much time as possible to make the modifications to the batch schedule. If adding or modifying batch jobs, ensure there are government personnel available to assist the contractor.
Frequency	Monthly
Measurement Techniques	The CTR will review the trouble tickets for requests and verify performance against the batch job monitoring reports.
Reports	1. Trouble tickets 2. Monitoring logs 3. Server logs 4. Batch job schedule
Person Responsible for Verification	The CTR is responsible for verification.
Escalation Procedures	The CTR will be notified of any threshold violations. The CTR will attempt to resolve all disputes. Disputes that cannot be resolved will be presented to the COR.
Contractual Exceptions	None

Penalties/Rewards	Minor penalty: 5% monthly rate • Any threshold values were exceeded. • Major penalty: 20 % monthly rate • Three minor penalties within the year • 11.2 If any of the maximum time frames designated in the batch job request section were exceeded.

Service Name	SLA 12.0 Technology Refresh Rates
Service Description	Technology is changing at a rapid pace. To take advantage of new innovations, technology must be updated. This SLA specifies the time frames for technology refresh rates. Technology refresh requires coordination between the government and the contractor. The coordinator cannot upgrade to a new version of system software or hardware without ensuring that the application is not affected. Conversely the government must ensure that if the application developers are designing new functionality that requires an upgraded hardware or a new version of software that the contractor is willing and able to support the upgrade.
Reason for Measuring	Technology needs to be updated on a consistent basis, not only to take advantage of the benefits offered by that technology, but for interoperability purposes as well. Technology refresh also allows software developers the opportunity to take advantage of the most recent scientific advancements.
Time Frame	Quarterly
Scope	Technology refresh applies to all hardware and software in the contractor's host environment that supports the application, including firewalls.
Performance Category	12.0 Software Refresh
Performance Metric	The contractor is responsible for the planning, installation, and testing of system and infrastructure software upgrades. New software will not be installed upon release. The contractor must have time to test the new version, and develop an installation plan if the upgrade is extensive. However, the time from release to installation should be quick enough to allow the government to take advantage of any benefits, and to ensure interoperability. This SLA is concerned with the installation timeframes for new versions of software. Patches or upgrades to existing versions are covered under another SLA. The metric used will be the time from the release of the new software version until it is installed in an operational environment.
Threshold Levels	The following are the thresholds for software refresh: Essential: 18 months Enhanced: 12 months Premier: 6 months

	No system or infrastructure software will be more than 2 releases behind the most current software release.
Formula	None
Assumptions	Some legacy application software have dependencies that do not allow for system software upgrades. In the case of hard coded dependencies, only non-dependent software would be upgraded.
Contractor Responsibility	Notify the configuration review board of any software upgrades. This requires that the contractor keep abreast of latest changes in technology. It also requires that the contractor determine how the new changes will affect the hosted application. This will require testing and coordination with the government developers.
Customer Responsibility	Cooperate with the contractor in any functional tests required to test a new software release. The government developers should also be aware of and take advantage of the latest software releases.
Frequency	Quarterly
Measurement Techniques	The CTR will verify software refresh rates by reviewing recommendations from the vendor, minutes from the change review board, scheduled maintenance reports, configuration documentation, and spot-checking the latest releases with the applicable vendors.
Reports	1. Minutes from the Change Review Board 2. Scheduled maintenance reports 3. Configuration documentation 4. Software refresh recommendations from contractor
Person Responsible for Verification	The CTR is responsible for verification.
Escalation Procedures	The COR will be notified if there are any disagreements on interpretation.
Contractual Exceptions	None
Penalties/Rewards	Minor penalty: 5% monthly rate • Any threshold values were exceeded. Major penalty: 25 % monthly rate • Two minor penalties within the year

Service Name	SLA 13.0 Administration
Service Description	Administration is a general category that is concerned with ensuring documentation is up to date, accurate and is delivered in a timely manner. It also addresses attendance at required meetings and adhering to contractual procedures. The delivery of reports address the time frame that the various report deliverables must be delivered to the designated government representatives.
Reason for Measuring	Since many of the reports produced by the contactor are used to provide oversight of the contractor's performance, it is important that the reports are accurate and timely. Some reports are also used to perform quality control. If the information contained in those reports is delayed, potential corrective actions will also be delayed. Everyone's time is valuable. If a contractor is needed at a meeting, such as the configuration review board, it is important that a representative, with the appropriate power making authority attend, not only to represent the interests of the contractor, but also to ensure that the scheduled business can proceed.
Time Frame	The time frame is 24 X 7.
Scope	Delivery of Reports includes all the reports defined and agreed upon in the deliverables documentation. In addition to the reports defined in the deliverables document the contractor must also provide SLA compliance reports and associated reports that provide background, detailed information, or the raw information that may have been consolidated for the SLA reports. Delivery time frames are outlined in the statement of work or the corresponding deliverables section of the contract. Scheduled meetings refer to planned meetings that occur on a frequent basis, such as the configuration review board. It does not include short notice meetings that were not on the agreed upon meeting schedule. License management covers all software that is utilized in the contractor's host environment, including the application itself. Licenses for GOTS applications are not in the scope of this SLA. Change management procedures covers changes made to

	any software or hardware in the contractor's host environment, including the application. The contractor and the government will promulgate the change management procedures in a change management document that will be mutually agreed upon. This plan will discuss how the change review board will function, requirements for documenting the change, and testing requirements.
Performance Category	13.0 Delivery Schedule
Performance Metric	The contracted delivery time frames for the document deliverables will be evaluated against the actual delivery time.
Threshold Levels	The thresholds are as follows: Essential – Premier: Reports are due within one business day of their due date.
Formula	None
Assumptions	Government requests for reports that are not specified in the contract will go through the CTR for contract scope determination. If the contractor agrees, the request will be categorized as a priority 4 problem resolution and will require a trouble ticket from the contractor's help desk. Conflicts, or requests outside of the scope of the contract will be referred to the COR.
Contractor Responsibility	The government will work with the government representatives to determine the method of delivery. If there are problems, the contractor will contact the CTR for resolution.
Customer Responsibility	The government representative will work with the contractor to determine delivery methods and designate a primary and alternative receipt representative.
Frequency	Monthly
Measurement Techniques	The CTR will spot check documentation deliverables and determine when they were delivered. The contract will specify when the documents are to be delivered. The CTR will compare the delivery time designated in the contract with the actual delivery time to determine compliance with the thresholds. Actual delivery times will be determined by interviews, or the timestamp on documentation that has been e-mailed.
Reports	1. Hosting contract
Person Responsible for Verification	The CTR is responsible for verification.
Performance Category	13.1 Documentation Accuracy
Performance Metric	This measurement ensures the accuracy of the

	documentation that is delivered. For example, configuration data must be accurate and up to date for disaster recovery, testing, and software development purposes. It is not enough to simply deliver documentation; the information contained in that documentation must be timely and accurate.
Threshold Levels	The thresholds apply to all required documentation. Inaccuracy is a subjective determination made by the CTR. The document must contain more than three non-significant errors, or one significant error. The CTR will determine the criticality of the error with respect to its affect on the application and the business processes the application supports. Non-significant error would be addition errors that do not significantly affect the computational outcome, missing serial numbers on hardware configuration documentation, or fail to update equipment moves within the host environment. Significant errors would include failure to update the backup schedule with new systems, failing to update the software configuration documentation with new upgrades, or failing to produce installation procedures for a system. The thresholds for accurate documentation is as follows: Essential – Premier: 95%
Formula	The number of documents audited with no errors divided by the number of total document deliverables.
Assumptions	The CTR will be able to determine whether a problem is significant or not. Discussions with the program manager and the contractor may help to categorize the severity of the document oversight/error.
Contractor Responsibility	The contractor will determine the root cause of any documentation errors, and attempt to automate as much reporting as possible.
Customer Responsibility	The CTR will inform the contractor of any errors discovered in the documentation.
Frequency	Monthly
Measurement Techniques	The CTR will perform spot checks on the documentation. Most errors in the documentation will be discovered through problem resolution, red team vulnerability assessments, and configuration audits.
Reports	1. All required documentation is subject to audit. 2. Red team vulnerability assessments

Person Responsible for Verification	The CTR is responsible for verification.
Performance Category	13.2 License Management
Performance Metric	It is illegal to operate third party software without proper licenses. The contractor is responsible for ensuring that all software that is a part of the host environment is supported by valid licenses. License management also includes the application and it's associated databases.
Threshold Levels	All software must have current licenses. Shareware and freeware can be utilized in accordance with the acceptance agreements related to the specific software. The threshold for proper licenses are as follows: Essential – Premier: 95%
Formula	None
Assumptions	Government Off the Shelf (GOTS) software will not have to have a license.
Contractor Responsibility	The contractor must have a process in place to ensure that all software in the host environment, including the application, has valid licenses. If the license is based on the number of concurrent users, the contractor will be responsible for ensuring the users do not exceed the license agreement. The contractor will notify the government of licenses about to expire, as well as when licenses need to be renegotiated to support an expanding user base.
Customer Responsibility	Copies of all license agreements must be turned over to the contractor before the software can be utilized.
Frequency	Quarterly
Measurement Techniques	The CTR will conduct spot checks of the licenses against the software configuration documentation.
Reports	1. Software configuration documentation 2. Software licenses
Person Responsible for Verification	The CTR is responsible for verification.
Performance Category	13.3 Meeting Attendance
Performance Metric	The contractor must have a representative at all scheduled meetings. The contractor would not have been invited to the meeting if the business did not involve the contractor. Participation is necessary to ensure that time is not wasted waiting for contractor input, or decisions from the contractor. All contractor representatives are expected to be able to represent the contractor and make decisions.

	The contractor and the government must develop a schedule for the meetings that the contractor is expected to attend. Meetings other than those agreed upon in the schedule of meetings will not apply to this SLA.
Threshold Levels	Thresholds for attending scheduled meetings is as follows: Essential – Premier: 95%
Formula	The number of meetings with a contractor representative in attendance divided by the total number of scheduled meetings.
Assumptions	The contractor will make every effort to attend meetings that were not in the official schedule. If enough warning is given, meetings will be rescheduled. Rescheduling of meeting should be coordinated with the program manager's staff.
Contractor Responsibility	Ensure the individual attending the meeting has the ability to represent the interests of the contractor as a voting member.
Customer Responsibility	The customer must determine which meetings the contractor needs to attend. Once the meetings have been identified, then the government must work with the contractor to develop a schedule that both parties can agree to. The government will notify the CTR if the contractor has failed to attend any scheduled or rescheduled meetings. A copy of the notification will be sent to the contractor. It is not necessary to notify the CTR of rescheduled meetings.
Frequency	Monthly
Measurement Techniques	The government will notify the CTR and the contractor when the contractor has failed to attend a scheduled meeting. If there are any challenges from the contractor, the CTR will compare the schedule of meetings against the minutes for those meetings. The meeting minutes will contain the attendees. If no contractor representatives were in attendance, then the challenge will not be accepted. If in the opinion of the CTR and program manager, the contractor has provided enough warning to reschedule a meeting, that particular meeting will not be counted in the SLA computations.
Reports	1. Meeting schedule 2. Meeting minutes 3. Notification from the government of missed meetings

Person Responsible for Verification	The CTR is responsible for verification.
Performance Category	13.4 Change Management Processes
Performance Metric	Before a software or hardware change (modification, upgrade, new version, updated hardware, etc…) is implemented, it must first be approved by the change review board. Maintaining control of software and hardware configuration changes is essential to the ensuring architectural conformity, disaster recovery, compatibility with other software, interoperability, and quality assurance. The metric will be the percentage of hardware and software changes that were executed in accordance with the change management procedures.
Threshold Levels	The thresholds for abiding by the change management procedures is as follows: Enhanced – Premier: 95%
Formula	The number of hardware and software changes that were executed in accordance with the change management processes divided by total number of changes executed.
Assumptions	All change review board meetings are documented to capture those changes that have been approved, and disapproved. All configuration changes will be documented.
Contractor Responsibility	Ensure change management procedures are followed. If changes are needed before the board can convene, the contractor will work with the government to gain approval.
Customer Responsibility	The government must hold change review boards often enough to support change requirements. If changes are occurring at a rate that is not supported by the change review boards, the government will appoint a representative to review and approve urgent changes.
Frequency	Monthly
Measurement Techniques	The CTR will review the configuration documentation and the system and monitoring logs to ensure that only approved changes were installed on the system.
Reports	1. System logs 2. Configuration documentation 3. Change Review Board Meetings 4. Monitoring logs
Person Responsible for	The CTR is responsible for verification.

Verification	
Escalation Procedures	The COR will resolve any disputes regarding contractual interpretations, or categorization of document errors.
Contractual Exceptions	None
Penalties/Rewards	Minor delivery penalty: No monetary penalty • Any threshold values were exceeded. Minor accuracy, attendance, and change management penalty: 5% monthly rate • Any threshold values were exceeded. 13.0 Major delivery penalty: 10 % monthly rate • More than 3 minor penalties within the year • Daily reports exceeding 3 days • Weekly reports exceeding 4 days • Monthly reports exceeding 7 days • Quarterly reports exceeding 7 days • Annual reports exceeding 10 days 13.1 Major accuracy penalty: 20 % monthly rate • More than 3 minor penalties within the year • More than 1 significant error in one month 13.2 Major license penalty: 25% monthly rate • Any threshold values were exceeded. 13.3 Major attendance penalty: 10% monthly rate • More than 3 minor penalties within the year • Less than 80% attendance in one month 13.4 Major change management penalty: 20% • More than 3 minor penalties within the year • Less than 80% adherence to the policy in one month Any malicious or intentional inaccuracies in required documentation directly affecting SLAs may result in termination of the contract.

APPENDIX B

A. PURPOSE OF QUESTIONNAIRE

The purpose of this questionnaire is to determine if the readers believe that the use of service level agreements can improve software quality and post-production support for applications.

B. INSTRUCTIONS

This questionnaire consists of four sections. The first section is a brief background discussion on how service level agreements can contribute to software design and post-production support. The second section discusses the format of an effective service level agreement. The third section is a case study illustrating a real world scenario along with a service level agreement for availability. The last section consists of a questionnaire. Each statement has a corresponding Likert scale from 1 to 5, with a 1 representing strong disagreement and a 5 indicating strong agreement.

C. INTRODUCTION

Information technology has become pervasive in our daily business. The rapid growth of the Internet has lead to an increased reliance on interconnected computer systems to provide critical operational services from business processes to coordinating decentralized command and control systems.

As advances in technology encourage the adoption of new ways of conducting business, management and end users have become increasingly reliant on the underlying technology. Systems that used to be managed by functional experts are now totally reliant upon information technology to function. These business critical, IT intensive systems are becoming more complex, and difficult to manage, yet the performance expectations from management and the end-users continue to increase.

Unfortunately, despite software's increased importance to organizations the quality of software is still lacking. There are numerous examples of software errors leading to major incidents, including the Denver airport baggage handling system, the Ariane 5 explosion, the Mars Sojourner, and the Mars Climate Orbiter.

D. CHALLENGES IN OBTAINING QUALITY SOFTWARE

In his article "Why Software is so Bad", Charles Mann offers a number of reasons why the quality of software tends to be poor. Mann states that software quality is actually getting worse rather than better, despite the advances in software engineering theory, processes, methodology and tools. Poor software quality can be attributed to the following:

- The perceived need to hurriedly develop and market a software-based product to be the first to market; such an approach can result in software artifacts that contain software flaws and are difficult to test and maintain.

- Software is generally poorly designed. This is due in part to the poor training programmers have received, and the fact that as programmers bounce code off of the complier to fix errors, they often deviate from the original designs and end up with sloppy, poorly documented code.
- Testing software often requires a different skill set that programming. Often the personnel are not properly trained, or are not given the time to test properly.
- Software is not designed for testing. The designers did not utilize component level design or software architecture, the software's modularity and corresponding interconnectivity was not well defined, and the application was not internally coded to throw exceptions, or write faults to a log.
- Software fails to meet the customer's expectations. The software developer must looks at requirements from the user's perspective, the business' perspective, and the programmer's perspective. Too often the user is not a part of the requirement elicitation process.
- Requirements churn contributes to the poor reliability of software, as designs are altered, interfaces added, unplanned modules are glued together, with little consideration given to the additional resource constraints.
- Post-production support plays a large role in the success of an application, but the software developers do not normally address it in their planning.
- The application needs to be hosted in an environment that supports the application's functionality. Software quality can be adversely affected by lack of resources within the server, and by network and bandwidth constraints.
- Maintaining software without proper documentation or configuration information is very difficult and expensive. Additionally, without proper documentation it is difficult to compare the original requirement specifications to the product throughout the software's lifecycle.

There have been a number of initiatives proposed to improve the quality of software through its lifecycle. Most approaches believe that quality must be designed into a product. Approaches such as formalizing specifications, use of development standards and models, and utilizing architecture for quality analysis support this approach. Others believe that the answer lies in creating languages that are designed to prevent common errors such as Ada, or utilizing rigorous testing and third party debugging tools.

If there are numerous approaches to developing quality software, why are there still problems? Part of the answer lies with the lack of meaningful dialog between the developers, end-users and management. Unrealistic completion dates, requirements churn, poor requirements elicitation, and lack of proper resources all lead to development problems. Additionally, just because standards exist for developing software does not mean that they are being used. In many cases adherence to developmental standards requires additional training, additional development time, and additional funds.

E. SLAS: WHAT THEY ARE AND HOW THEY ARE USED

One approach to improving software quality and post-production support is through the use of service level agreements (SLAs). Service Level Agreements (SLAs) have long been used as a contractual mechanism to specify the means to measure whether requirements were performed as desired. SLAs specify the metrics to measure adherence to specific requirements (usually contained in the Statement of Work). SLAs are traditionally used with outsourcing contracts, but more organizations are using them internally to measure the level of service that the IT department is delivering. In his article Mann advocates the use of litigation to force organizations to develop software in a more responsible manner. SLAs are contractually binding, and can be used in a similar fashion, but without the need for excessive legislation.

SLAs are typically written from the end-users perspective and represent what levels of service or performance are acceptable to the end-user and what is attainable by the developer or provider. However, the levels of performance identified in the SLA must also ensure that the underlying business processes are supported. To ensure that all perspectives are taken into account, teams are normally formed to develop SLAs. The various stakeholders are represented and the levels of performance are identified, quantified, and agreed upon. The team must resolve a number of issues such as determining the business impact of the various level of service need to be identified, identifying metrics that are meaningful and measurable, assessing technical capability, identifying costs associated with the various levels of service, determining benefits of the service, and the team must develop SLAs that are agreeable to all of the stakeholders. The group development of SLAs help the various stakeholders understand each others bias, viewpoints, concerns, terminology, and perceptions. That understanding is essential in requirements determination.

Service level agreements assist in the development of quality software and post-production support in the following ways:

- Involving the end-users and business process owners in the SLA development process helps to better define requirements by converting non-functional requirements such as performance into quantitative metrics.
- Incorporating quality metrics into the SLAs ensures that the developers are focusing on quality early in the development process, where it can be effective.
- Developing the SLAs forces the team to evaluate the constraints on the project in terms of personnel, resources, funds, technical capability, and time.
- SLAs define specific performance parameters that are required to support a business process. As such, every SLA performance requirement must be analyzed, and validated to ensure that they are meaningful, cost-effective, and that they add to improving overall performance.
- SLAs help institutionalize a change review board to continually review the SLAs, evaluate new requirements and ensure maintenance actions do not affect the SLAs. The change review board not only ensures that changes are tested against performance thresholds, but they can also be used to ensure the changes conform to architectural constraints, and that they are properly documented.

- SLAs also require monitoring to ensure that quality standards or thresholds are being adhered to. Monitoring the application and host environment provides feedback on performance quality and identifies areas that may need improvement.
- Monitoring of the network, hardware, operating system, and the application not only assist in problem resolution, but trend analysis can indicate potential problems before they occur. Software quality cannot be measured without proper monitoring.
- By defining meaningful and measurable metrics in the SLAs, the end-users, business managers and programmers have realistic quantifiable requirements that can be used to determine architecture and design.
- SLAs set user expectations through defined performance levels. By explicitly stating acceptable performance levels, SLAs prevent expectation creep.
- SLAs help drive the managerial oversight to ensure that quality processes are adopted and adhered to.
- SLAs concerning application availability drive numerous quality initiatives in both the design and post-production support. Reliability constraints may drive code reuse, application monitoring, complexity analysis, extensive testing, efficient problem resolution procedures, a good backup plan, and disaster recovery.
- SLAs concerning maintainability could drive a well-defined and documented architecture that would promote design consistency through guidelines and design patterns, as well as accurate configuration management.
- SLAs can be drafted to include numerous security issues such as protocols and ports utilized, interface with third party products, encryption, VPNs, and tunneling.

SLAs can play an important role in addressing software quality. SLA thresholds drive many of the quality solutions that were discussed previously. SLAs established early in the development cycle can be incorporated into the overall design and architecture. SLAs applied to the post-production phase, such as security SLAs ensure the application is being supported in a quality host environment. SLAs carry sufficient weight through penalties and incentives to focus management attention on quality issues.

F. SLA FORMAT

Service level agreements have many formats depending upon how they are used. Internal SLAs between management and the IT department can be more informal because many of the procedural issues are stated elsewhere. SLAs involving external service providers need to be more formal.

SLAs serve as a mechanism to notify all parties of services that will be performed, performance expectations, responsibilities of all parties, penalties for non-performance, and SLA resolution procedures. SLAs also define the oversight and interaction between the program managers and the service provider.

SLAs are often used in conjunction with a Statement of Work (SOW), which provides the actual requirements. The SLAs provide the metrics to measure whether the requirements are being met. Most activities find it easier to keep the two documents separate, as many requirements will not have SLAs associated with them. SLAs should concentrate on the business critical measurements. The costs and managerial oversight

needed to track and verify whether SLAs are being met can quickly become overwhelming if too many SLAs are mandated.

G. CASE STUDY

A Navy activity has just completed a cost-benefit analysis study involving server consolidation and hosting services. They have decided to consolidate their servers and have them hosted by an external service provider (ESP). The ESP has state-of-the-art facilities, a highly knowledgeable staff, and can provide the needed services at a lower cost than the Navy activity is currently paying.

The same cost-benefit study recommended that the Navy activity retain responsibility for the application maintenance. It was decided that the current staff would be more responsive and flexible than a contractor. Additionally due to the complex reach-back issues with numerous legacy systems it was decided that the company could not loose the tacit knowledge the Navy employees possessed.

The Navy activity met with their current sysadmin staff, program managers, and contractors that they hired to advise them on hosting services. After numerous meetings, they generated the requirements they felt were necessary to support their applications. They also looked at the standard SLAs that the various External Service Providers (ESP) used. After review, they decided that the ESP SLAs were too vague, and did not provide the service levels they felt they needed.

The Navy activity then formed a group to generate SLAs for each application. The group consisted of various users, the program manager's staff, the business process owner, and participants from the various ESPs that were interested in participating. They decided to develop template SLAs that would provide the foundation that all of the applications would use, but that were easily tailored to meet the business needs of each application.

The Navy activity was also aware of the fact that they were soon going to go under the Navy/Marine Corps Intranet (NMCI). NMCI is a contract that the Department of the Navy has with EDS for desktop and infrastructure management. The Navy activity wanted to use end-to-end SLAs (measuring the performance from the client to the server), but could not, because the ESP did not control the client piece, the BAN/WAN, or connectivity from the ESP's Internet Service Provider to the servers. Additionally some of the applications were distributed and had to use the Internet to access data. As such, the team scoped the SLAs to include the host environment only.

The team developing the SLAs identified 14 service areas that they felt should be incorporated into the SLAs. The availability of the compute environment was the first service area they addressed. The team debated long and hard attempting to define availability. Since the application itself was the Navy's responsibility, the host environment then consisted of the operating system and monitoring software, the server, the host environment network, and the firewall. Ultimately they decided that availability should be defined in terms of an application's ability to compute. Defining the availability in these terms not only captured hard downtime (i.e., system crashes), but it also allowed them to determine resource thresholds that would impact the application's performance (soft crashes where application performance was degraded enough that it did not produce the desired utility). This approach was needed since response time

measurements from the client to the server were not meaningful, as the service provider did not control the entire infrastructure.

The team felt that other service areas that should be covered by SLAs are restoration of service, help desk services, problem resolution, request management, security management, software maintenance, maintenance schedules, migration service, backups, batch services, technology refresh rates, administration, and customer satisfaction. Many of the service areas had sub-sections that dealt with specific areas within the larger service area. For example, help desk services included sub-sections for help desk availability, initial feedback (monitoring time when user is informed the trouble ticket was received and an estimated resolution time was given), accuracy of problem resolution, customer satisfaction, accuracy of trouble ticket reports, and occurrence of repeat problems.

The SLA that the team developed for host environment availability is presented below. The intent was that this SLA would serve as a template for other applications. The SLA would cover most applications, but the SLAs could be modified if needed.

H. SAMPLE SLA

Service Name	SLA 1.0: Compute Service Availability
Service Description	Availability measures the capability of an end-user to access and fully utilize an application (according to specifications) over a period of time. Availability is usually expressed as a percentage of time that the system was available for use divided by the agreed upon hours of operation. The time period that an end-user cannot utilize the application is considered 'downtime'. Availability metrics are generally intended to be end-to-end, reflecting availability from the end users perspective. However, these SLAs only cover the host environment, so availability metrics will be restricted to the host environment only, and will not apply to the client piece or the connectivity from the client to the host environment firewall. Downtime can also be difficult to define. This SLA will concentrate on an application's opportunity to compute. The thresholds will contain metrics to ensure that the application has sufficient resources to operate to specifications. If the compute environment is not operating at a certain level of efficiency, the application performance suffers. As a result, if certain resource thresholds are not met, the period of time the resources do not meet the thresholds will count as downtime.

	Response time is another element of availability that must be addressed. The SLA is limited to the host environment, so application response time will be calculated from the time a server receives application input until it provides the correct output. It is necessary to develop a program that resides on the server in order to generate the information necessary to measure response time (this is often referred to as synthetic transactions). The program will test key application functionality at random times and measure the response time from when the input is initiated until the desired output is correctly received. Response times will apply to premier services only. Development of the program will be negotiated as a separate line item if the program wants the service provider to perform that function.
Reason for Measuring	Availability is a measure of quality. The program manager and the contractor need to constantly monitor the infrastructure, hardware and system software to measure the effectiveness of the hardware and software in supporting the application. Diligent monitoring will detect early signs of problems that may require maintenance action. The efficacy of the application support has direct business impacts. When the application is not available any business related to that application stops; opportunities are missed, business processes are impacted, and deadlines can be missed. The program manager must identify a target availability threshold and be able to justify expenses associated with it. This will involve determining the business impact of lost service. The contractor must evaluate the infrastructure to determine if it is possible to support the availability, or if redesign or additional redundant or high availability equipment is needed. The host environment cannot be designed, implemented, or managed unless an availability threshold is established.
Time Frame	Derived by the contracted number of support hours. The Default is 24x7x365. Scheduled maintenance time that is within the maintenance window, and does not exceed the agreed upon maintenance time frames will not be included in availability computations.

	Additionally, scheduled maintenance involving the application (i.e., granting root access to maintenance personnel to perform an upgrade) will not be considered down time. The Maximum "Available" time will be determined from the hours of support that were contracted. Example (1): Hours of Support = 24 x 7. The maximum "available" time in a 30 day month is 30 x 24 x 60 = 43,200 minutes. Example (2): Hours of Support = 9 x 5. The maximum "available" time in a month with 21 work days is: 21 x 9 x 60 = 11,340 minutes.
Scope	This is an end-to-end metric from the host environment firewall to the application. It includes the hardware and the software for the firewall and server farm network, in addition to the hardware and software necessary to support the application. It does not apply to the application itself.
Performance Category	1.0 Host Environment Availability
Performance Metric	Availability is expressed as a percentage of the time that an application is fully functional divided by the total time encompassed in the support hours.
Threshold Levels	Availability thresholds are as follows: Essential Services: 99.50% Enhanced Services: 99.90% Premier Services: 99.95% In this SLA, availability is not only dependent upon the individual components that comprise the infrastructure (servers, network and firewall); it also addresses application and data availability from a security perspective. The following thresholds apply to resource utilization and network efficiency. If these thresholds are violated, then the application is considered 'down', and will count against availability: <u>Server Measures:</u> CPU Utilization: 80% sustained for over 1 hour. Not to exceed 90% for more than 2 polling cycles (5 minute intervals). Frequency of Failure: More than 3 service interruptions in

	one day. CPU run queue length: 3 Disk Utilization: 90% Disk Response Time: .25 second Disk Average Queue Length: 3 Disk I/O rate: 100 ms avg (Specific to hardware and configuration). Swap space availability: 90% of defined space Memory paging: 5 per second <u>Network Measures:</u> Data Delivery Rate: 99.95% LAN Latency (one way): 70 ms LAN Packet Collisions: More than 7% of packets transmitted (average based on 1 hour interval). Bandwidth Availability: 85% of defined bandwidth Ethernet Segment Utilization: Less than 30% <u>Security Related Measures:</u> If application performance is degraded due to an intruder attack, virus, worm, or security breaches previously identified, the application will be considered “down”. This includes the time that the application is affected during efforts to correct the violation. New attacks that have no previous history or signature will not be counted as “down time” against availability. Response Time: Will be depended upon the types of transaction that are being performed. If all transactions are similar, one threshold value can be determined, if they are all different, the thresholds should be specific to the transaction. The time to generate a simple report might be 1 second. Another area that we may want to include in the availability SLA is frequency of failure. This represents service interruption. All hardware errors affecting the application are considered ‘downtime’, and will be counted against availability. Application response time is dependent upon the type of functionality that is processed by the application. The SLA will specify the key functional processes and the

	corresponding response time expected. The processes and response times will be negotiated on an application-by-application basis.
Formula	Availability = (total uptime minutes) / (total uptime minutes + total downtime minutes) * 100
Assumptions	Downtime starts with the generation of a trouble ticket, or when a threshold violation is captured by the monitoring tools. Problems relating to the firewall, network, server or system software will count towards downtime. A review of the trouble tickets will verify that the downtime is properly assigned. Downtime attributed to application errors will not be included in the computation. Downtime that is a direct result of government actions will not be included in the computation. An example would be rebooting the system following an application update. Errors attributed to the client side portion of the compute environment will not be charged against the server farm reliability calculations.
Contractor Responsibility	Adopt and implement an industry-standard software solution for automatically polling and calculating compute service availability. Monitor compute services for earliest identification of outages. Take appropriate actions to correct deficiencies.
Customer Responsibility	The customer is responsible for prompt notification of any suspected compute service outages.
Frequency	Monitoring is conducted during scheduled support hours. Report frequency is monthly.
Measurement Techniques	The server will be ’Pinged‘ from a management server every 5 minutes. Failure by the server to respond will start the service outage time. The time between the first ’Failed‘ Ping and the first successful Ping after repair will be reported as Downtime. Approved industry standard monitoring tools such as Tivoli® and Open View® will be used to monitor resources. Operating system logs will also be used to determine compliance.

	Critical threshold violations will be considered downtime. Violations will be considered when a threshold is violated for three consecutive monitoring cycles. Example: Server A polled at 10:40, 10:45 and 10:50 and does not respond to the 10:45 poll but does respond at 10:40 and the 10:50. This would be calculated as 5 minutes of downtime. The downtime will be reviewed and adjusted by a contractor representative to exclude all outages from maintenance windows or outside the scope of service: • All planned outages • All outages due to application failures Adjusted Compute Service Availability is then recalculated. The new formula would be as follows: Availability = (total uptime minutes – downtime outside of scope) / (total uptime minutes – downtime outside of scope + total downtime minutes) * 100 Example Calculation: Server contracted for 7 x 24 hour support. Two outages occurred during a month with 30 days: (1) 100 minute application outage and (2) a 360 minute system failure occurred for a total downtime of 460 minutes. Availability is reported as: Reliability = (43,200 – 100) / ((43,200 – 100) + 360) * 100 = 99.17%
Reports	1. Monitoring Reports: Weekly 2. Trouble Tickets: Weekly
Person Responsible for Verification	The CTR will be responsible for reviewing the monitoring reports and trouble tickets to determine compliance with the SLAs.
Escalation Procedures	The CTR will be notified if the application is not accessible or functioning by the following time frames: Essential Service – after 30 minutes Enhanced Service – after 15 minutes Premier Service – after 10 minutes If there are any disagreements concerning whether downtime should be charged to the application, or the host

	environment, the CTR will make the decision. Disagreements can be escalated to the COR.
Contractual Exceptions	Availability does not include scheduled maintenance downtime within the maintenance window.
Penalties/Rewards	Minor penalty: 10% of monthly rate • Threshold values exceed agreed upon rates. Major violation: 25% monthly rate • More than 3 minor penalties during the year • Any availability less than the following: Essential Services: 98.0% available Enhanced Services: Target: 99.0% available Premier Services: Target: 99.5% available • More than 2 major violations will force escalation procedures between the COR and the contractor. Following escalation procedures additional missed targets may be cause for termination.

I. QUESTIONNAIRE:

1. How does your job relate to information technology?
   CIO Staff
   Software Developer
   SysAdmin
   Project Manager
   IT User
2. How many years have you been working in the IT field?
   0-2 2-4 4-6 6-10 Greater than 10

The following questions are based on a Likert scale. The scale is as follows:
1 = strongly disagree 2 = mildly disagree 3 = neutral 4 = mildly agree 5 = strongly agree
Annotate the number corresponding to your answer next to the statements.

3. Use of SLAs will improve software quality throughout the application's lifecycle.
4. Use of SLAs will improve software quality in the development stage.
5. Use of SLAs will improve the quality of hosting services for applications.
6. Use of SLAs will improve the maintenance of software.
7. Service level agreements will improve requirements determination.
8. Use of SLAs will improve software security.
9. Service level agreements will facilitate the development of a change review board.
10. Service level agreements will ensure rigorous reviews of software changes to ensure quality is maintained.

11. Service level agreements will improve configuration management.
12. Service level agreements will improve the management of IT intensive systems.
13. Service level agreements will help to ensure that the IT system supports its underlying business process.
14. Service level agreements help manage customer's expectations.
15. End-users are more willing to accept a system its performance parameters are well defined within a SLA.
16. Use of SLAs will assist in the source selection of potential service providers.
17. The format of the SLA was easy to understand.
18. The format of the SLA provided enough information to specify the means to measure whether a requirement was performed as desired.
19. The format of the SLA was detailed enough to determine services to be performed, performance expectations, and responsibilities of all parties.
20. The format of the SLA provides a template that could be easily modified to support any application.
21. The SLAs would make source selection of potential service providers easier.
22. The administrative burden of managing the SLAs would outweigh their benefit.
23. The difficulty in developing the SLAs would be too cumbersome for organizations.
24. SLAs would not be developed because the people with the knowledge base to develop the SLAs were outsourced, or are not available.
25. Enforcing penalty clauses/withholding incentives is too difficult.
26. Service level agreements will not resolve the quality issues associated with rushing software to market.
27. Comments: Please group comments into the following categories:
    a. Effectiveness of using SLAs in software acquisition.
    b. Usefulness of the SLA format.
    c. SLAs contribution to software quality.
    d. SLAs contribution to post-production support.
    e. SLAs contribution to lifecycle management.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C

1. How does your primary job function relate to information technology?

CIO Staff	20.9%
Software Developer	9.3%
System Administrator	18.6%
Project Manager	37.2%
IT User	14.0%

2. How many years have you been working in the IT field?

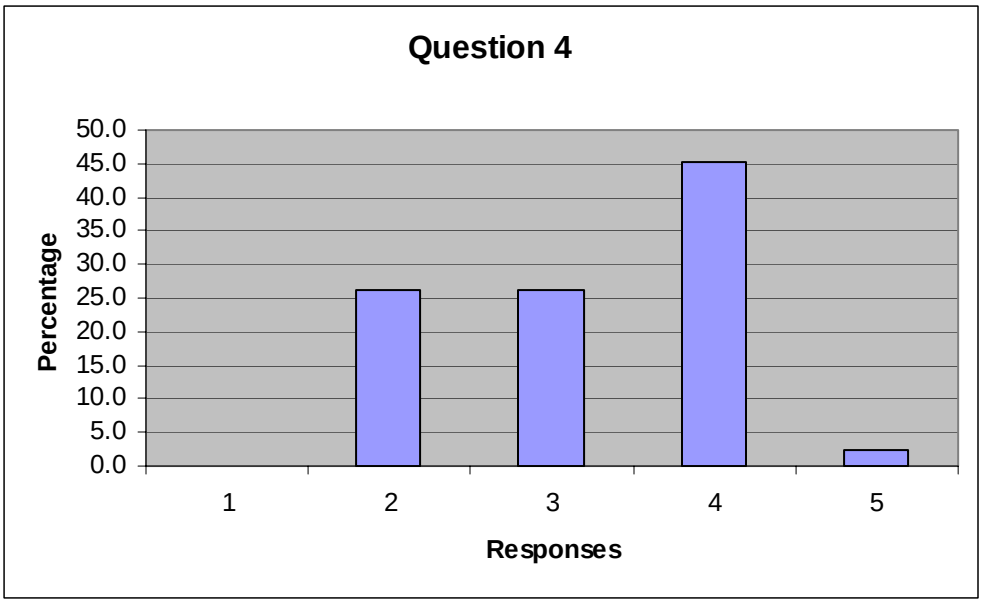
0-2	4.7%
2-4	4.7%
4-6	11.6%
6-10	20.9%
Greater than 10	58.1%

3. Do you have any previous experience working with SLAs?

No	32.6%
Less than 6 months	9.3%
6 Months – 1 Year	11.6%
More than 1 Year	46.5%

The following questions are based on a Likert scale. The scale is as follows:
1 = strongly disagree 2 = mildly disagree 3 = neutral 4 = mildly agree 5 = strongly agree
Annotate the number corresponding to your answer next to the statements.

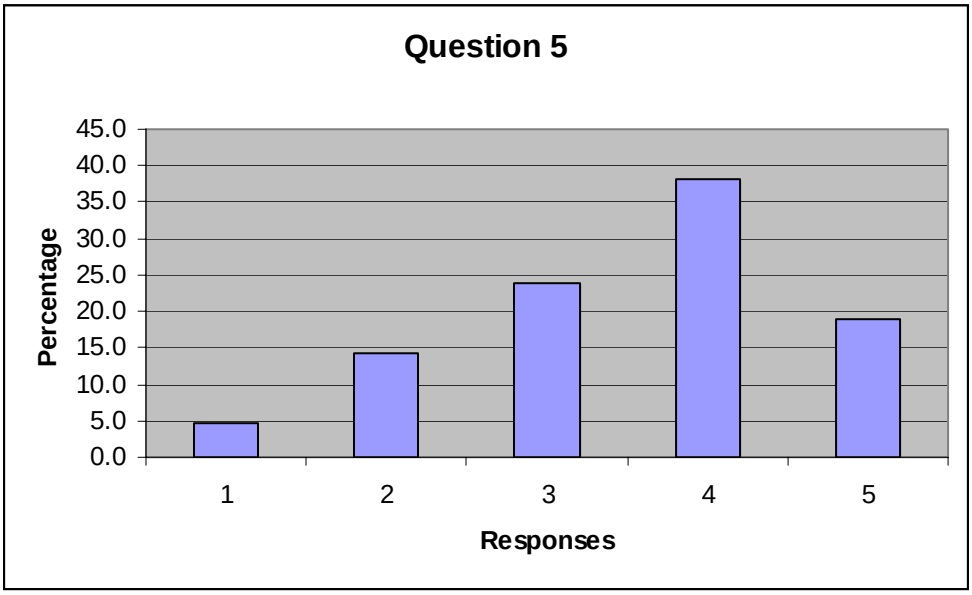
4. You are satisfied with the software quality in the applications you are familiar with.



Mean:	3.2381	Median:	3
Standard Deviation:	0.8782	Mode:	4
T-Value:	1.7571	P-Value:	0.0864

Statistically Significant: No

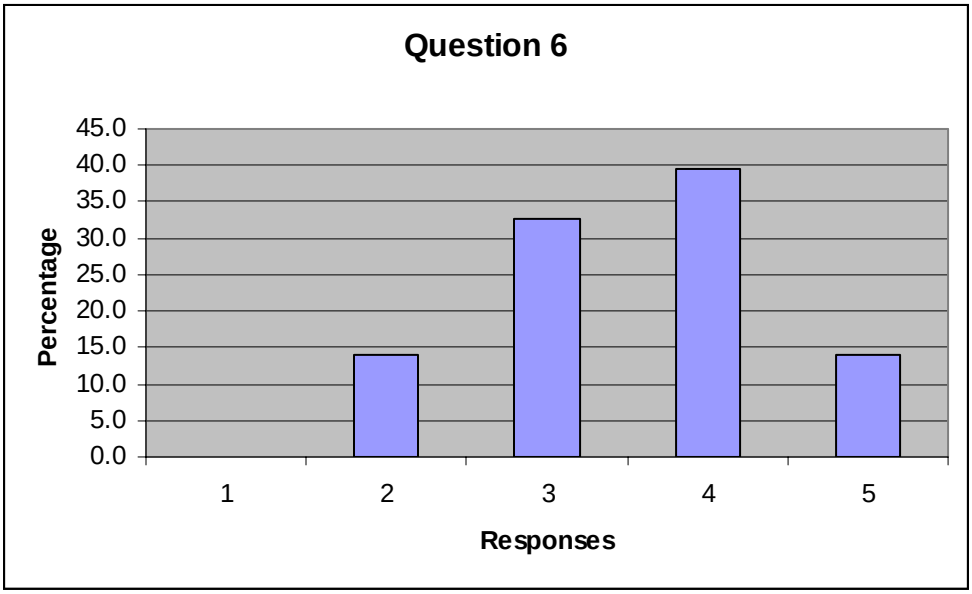
5. Management is concerned with post-production support of software.



Mean:	3.5238	Median:	4
Standard Deviation:	1.1096	Mode:	4
T-Value:	3.0595	P-Value:	0.0039

Statistically Significant: Yes

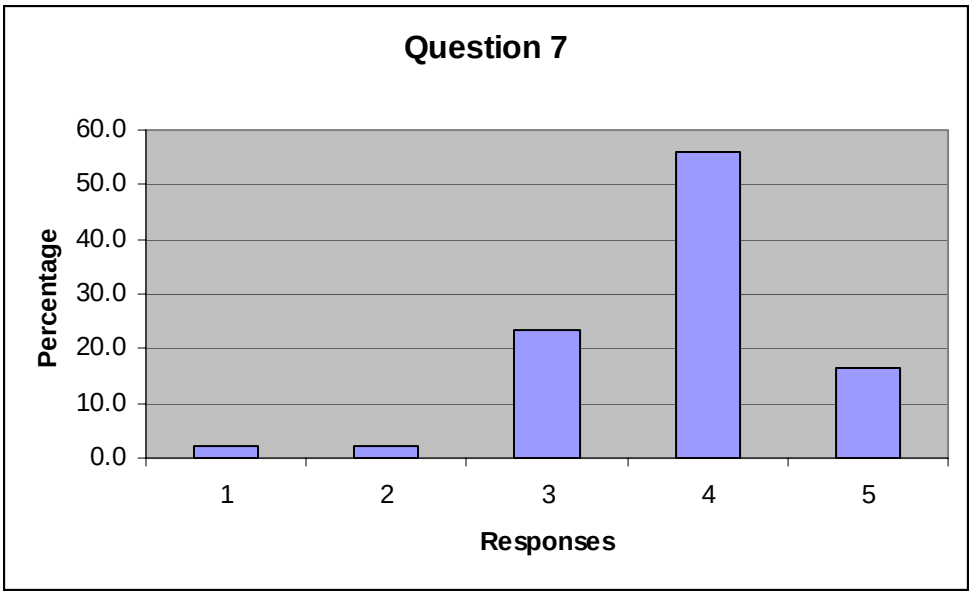
6. The format of the SLA was easy to understand.



Mean:	3.5349	Median:	4
Standard Deviation:	0.9089	Mode:	4
T-Value:	3.8589	P-Value:	0.0004

Statistically Significant: Yes

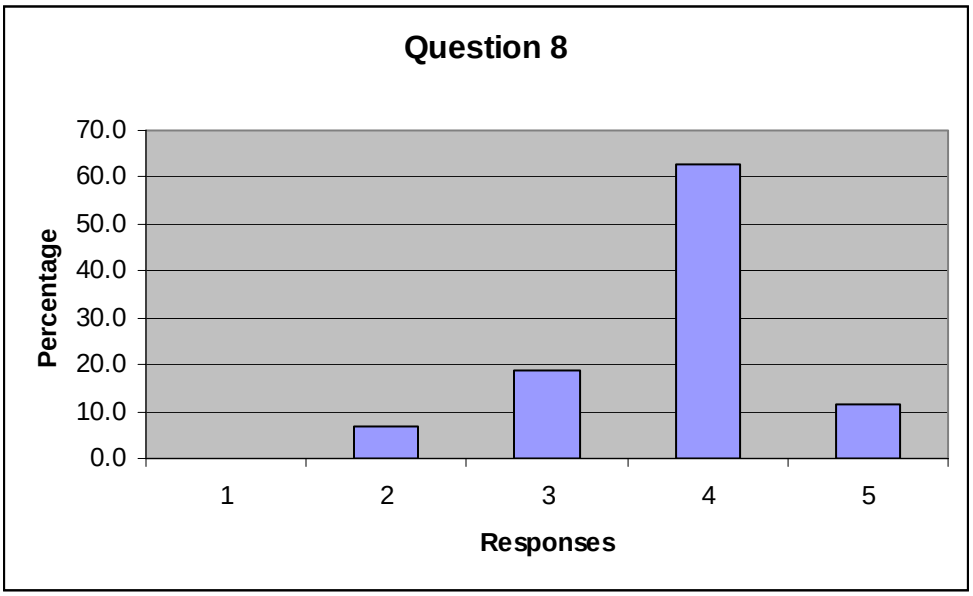
7. The format of the SLA provided enough information to specify the means to measure whether a requirement was performed as desired.



Mean:	3.8140	Median:	4
Standard Deviation:	0.8239	Mode:	4
T-Value:	6.4781	P-Value:	<0.0001

Statistically Significant: Yes

8. The format of the SLA was detailed enough to determine services to be performed, performance expectations, and responsibilities of all parties.

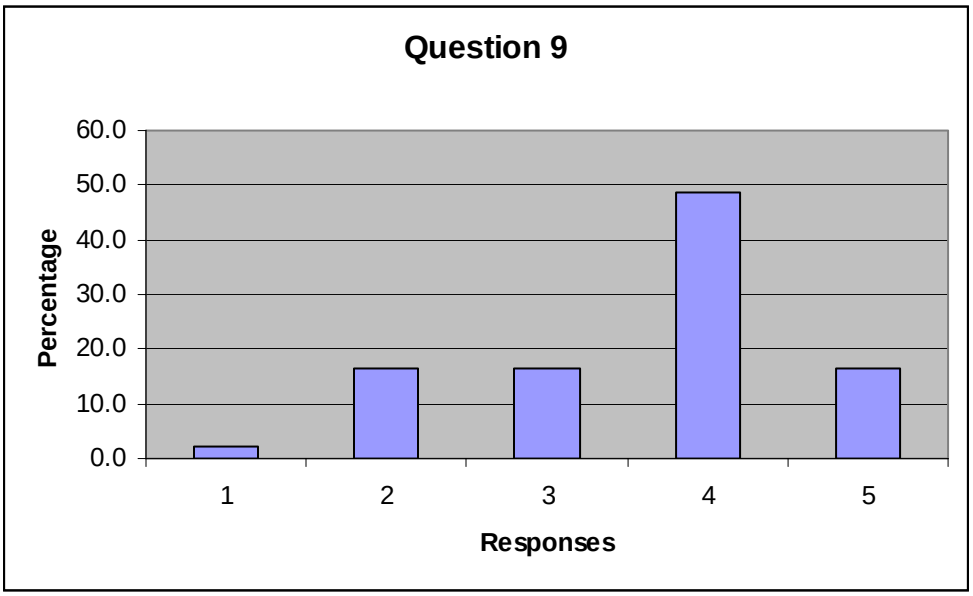


Mean:	3.7907	Median:	4

Standard Deviation:	0.7419	Mode:	4
T-Value:	6.9889	P-Value:	<0.0001

Statistically Significant: Yes

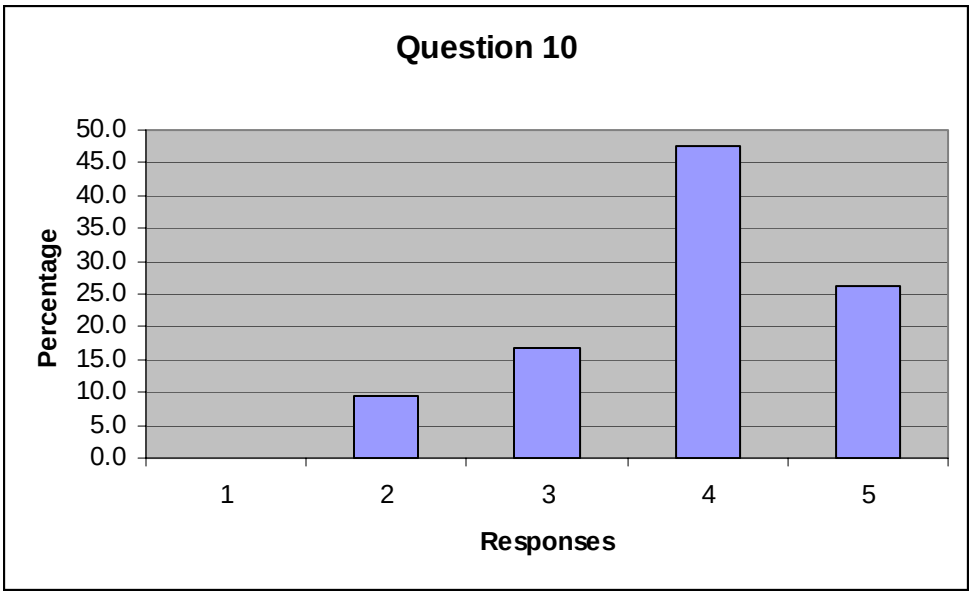
9. Use of SLAs will improve software quality in the development stage.



Mean:	3.6047	Median:	4
Standard Deviation:	1.0268	Mode:	4
T-Value:	3.8616	P-Value:	0.0004

Statistically Significant: Yes

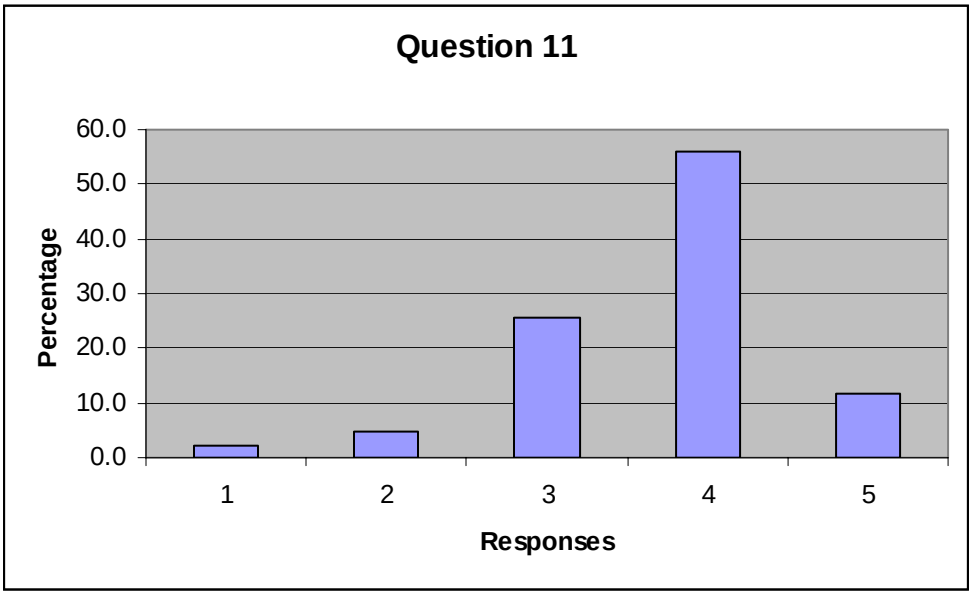
10. Use of SLAs will improve the quality of hosting services for applications.



Mean:	3.9048	Median:	4
Standard Deviation:	0.9055	Mode:	4
T-Value:	6.4753	P-Value:	<0.0001

Statistically Significant: Yes

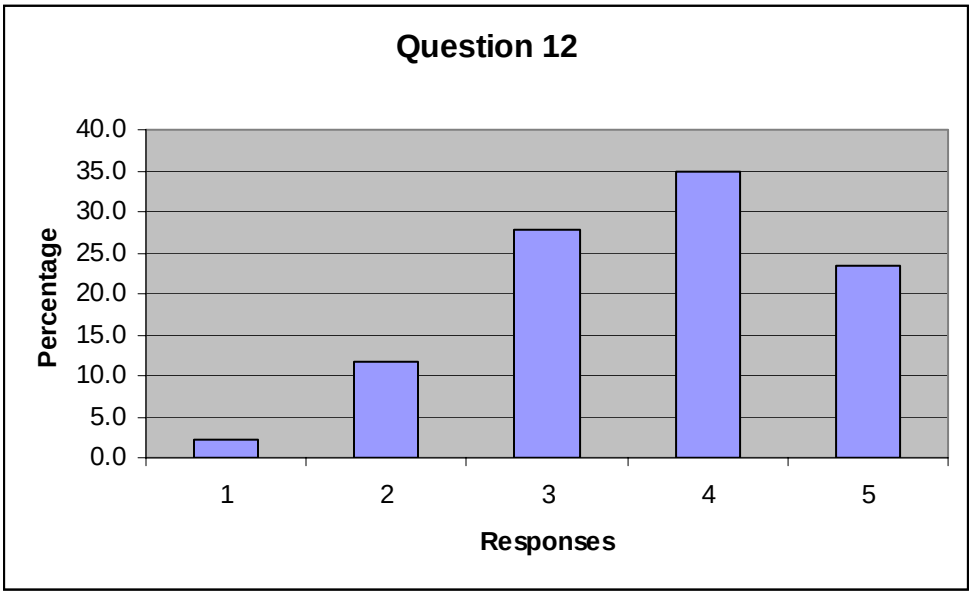
11. Use of SLAs will improve the maintenance of software.



Mean:	3.6977	Median:	4
Standard Deviation:	0.8319	Mode:	4

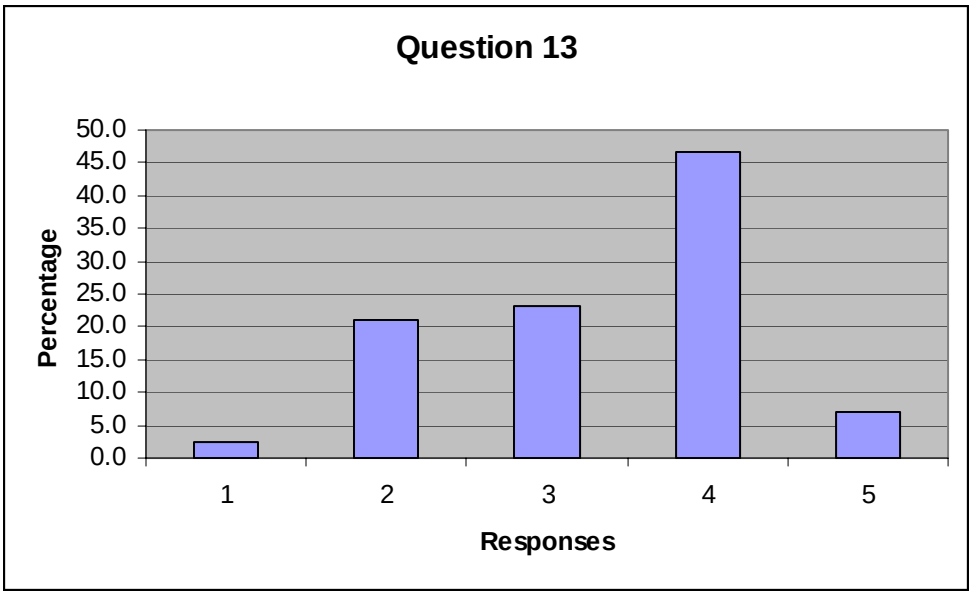
T-Value: 5.4991 P-Value: <0.0001
Statistically Significant: Yes

12. Service level agreements will improve requirements determination.



Mean: 3.6512 Median: 4
Standard Deviation: 1.0439 Mode: 4
T-Value: 4.0904 P-Value: 0.0002
Statistically Significant: Yes

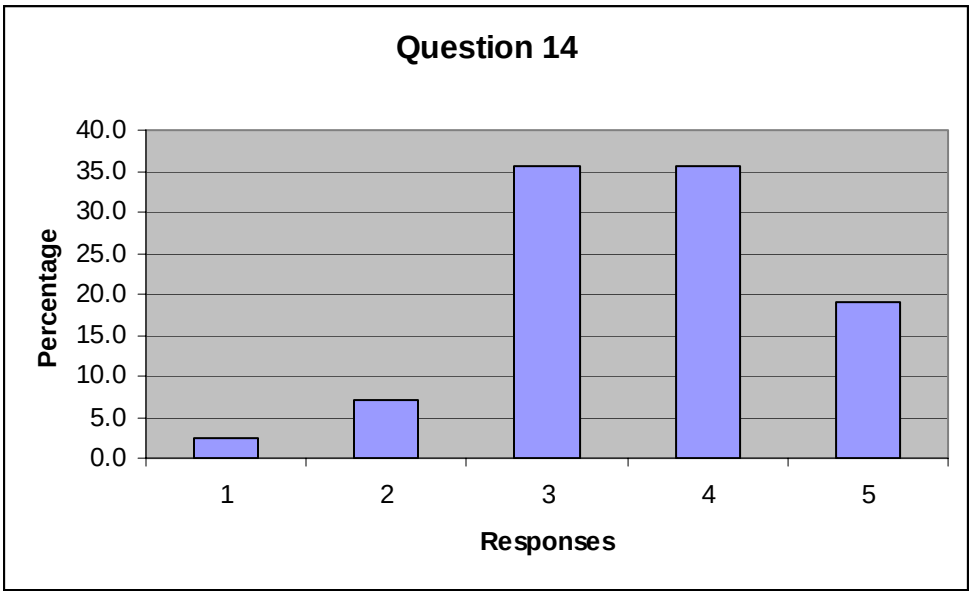
13. Use of SLAs will improve software security.



Mean:	3.3488	Median:	4
Standard Deviation:	0.9731	Mode:	4
T-Value:	2.3508	P-Value:	0.0235

Statistically Significant: Yes

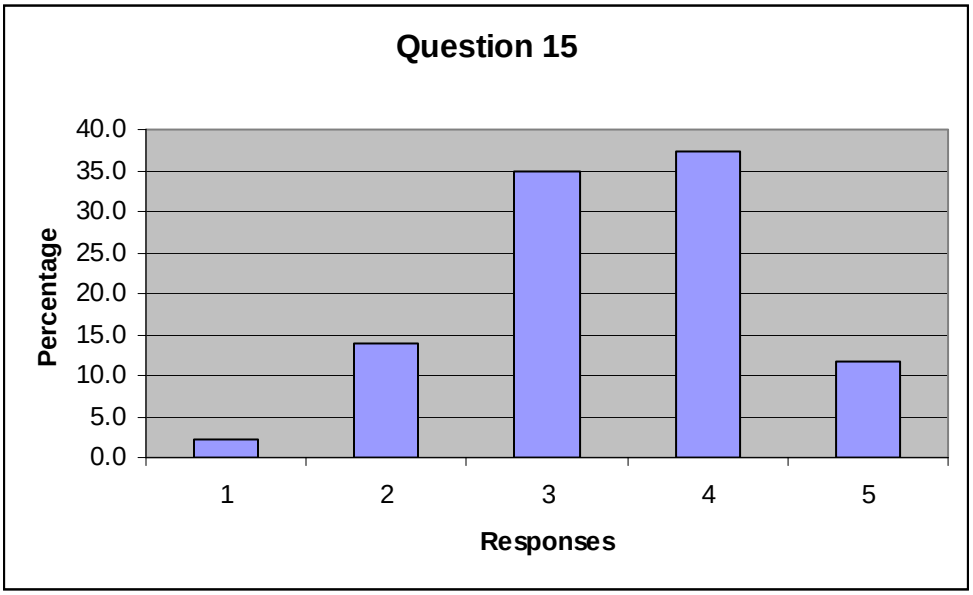
14. Service level agreements will facilitate the development of a change review board.



Mean:	3.6190	Median:	4
Standard Deviation:	0.9615	Mode:	3

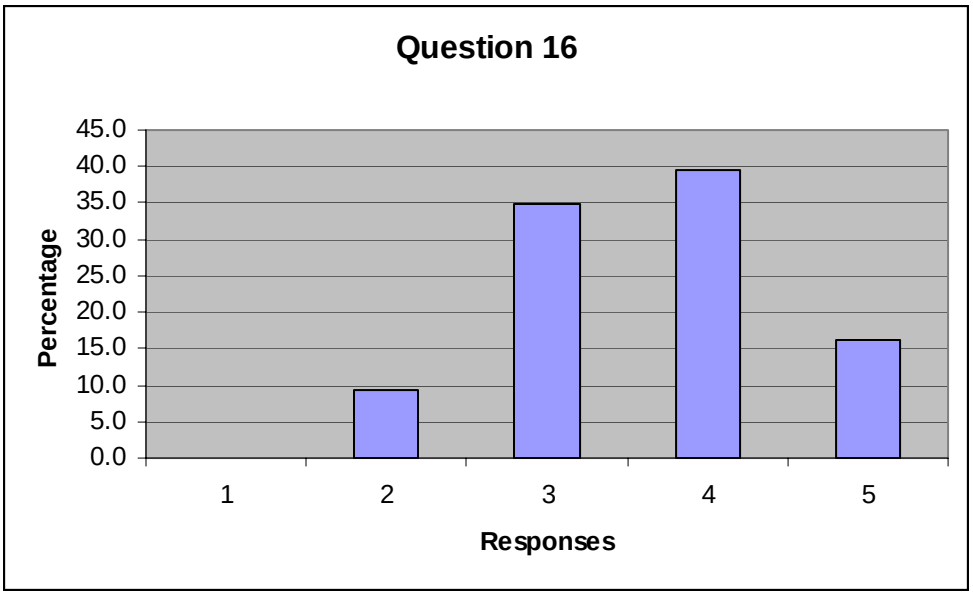
T-Value: 4.1725 P-Value: 0.0002
Statistically Significant: Yes

15. Service level agreements will ensure rigorous reviews of software changes to ensure quality is maintained.



Mean: 3.4186 Median: 3
Standard Deviation: 0.9570 Mode: 4
T-Value: 2.8683 P-Value: 0.0064
Statistically Significant: Yes

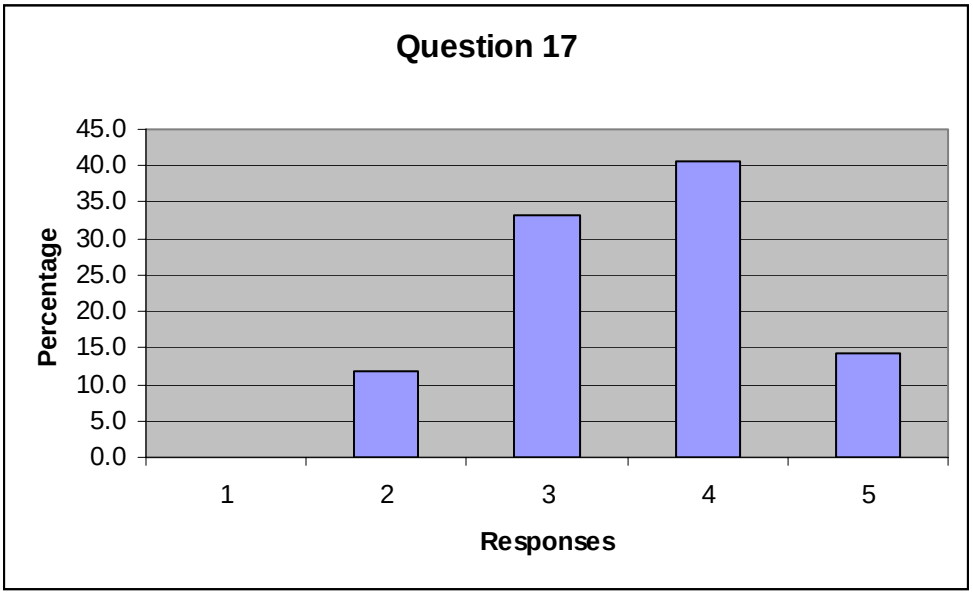
16. Service level agreements will improve configuration management.



Mean:	3.6279	Median:	4
Standard Deviation:	0.8735	Mode:	4
T-Value:	4.7137	P-Value:	<0.0001

Statistically Significant: Yes

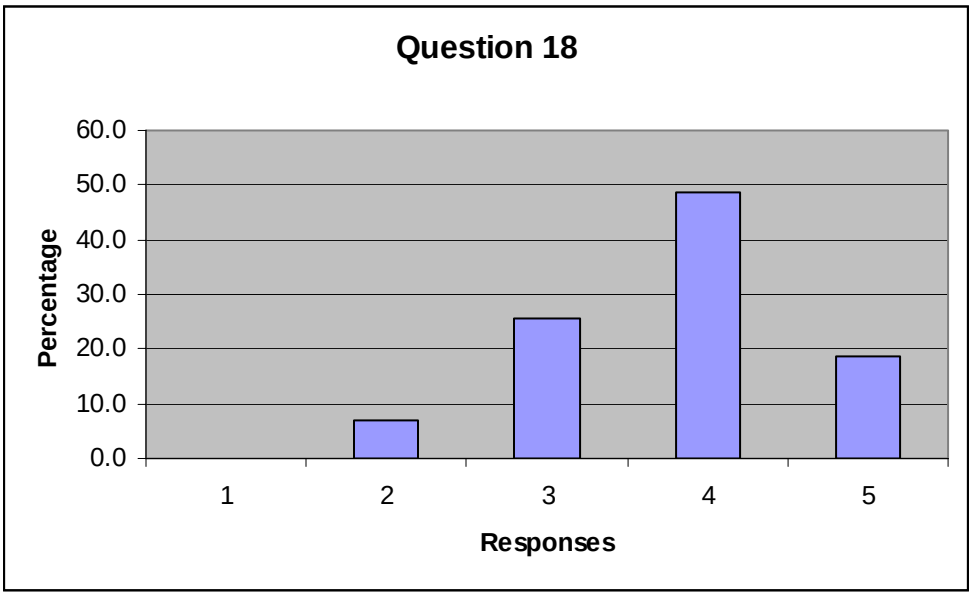
17. Use of SLAs will improve software quality throughout the application's lifecycle.



Mean:	3.5714	Median:	4
Standard Deviation:	0.8874	Mode:	4
T-Value:	4.1732	P-Value:	0.0002

Statistically Significant: Yes

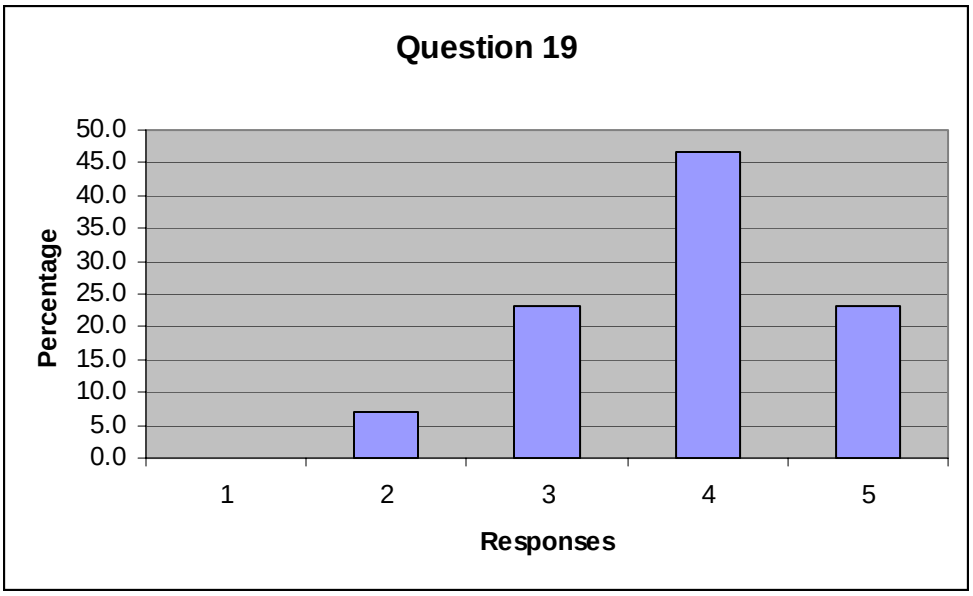
18. Service level agreements will improve software program management.



Mean:	3.7907	Median:	4
Standard Deviation:	0.8326	Mode:	4
T-Value:	6.2273	P-Value:	<0.0001

Statistically Significant: Yes

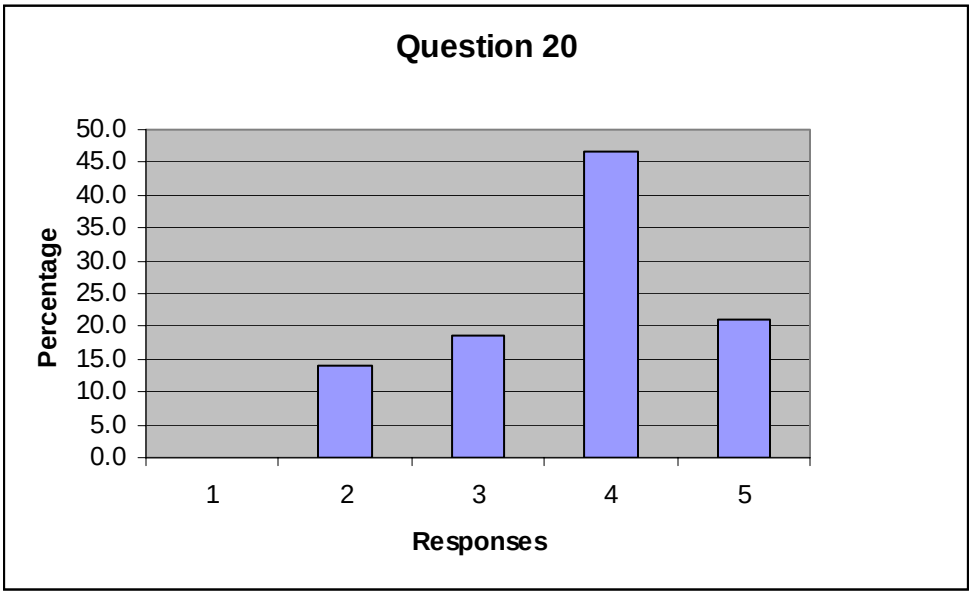
19. Service level agreements will help to ensure that the IT system supports its underlying business process.



Mean:	3.8605	Median:	4
Standard Deviation:	0.8614	Mode:	4
T-Value:	6.5505	P-Value:	<0.0001

Statistically Significant: Yes

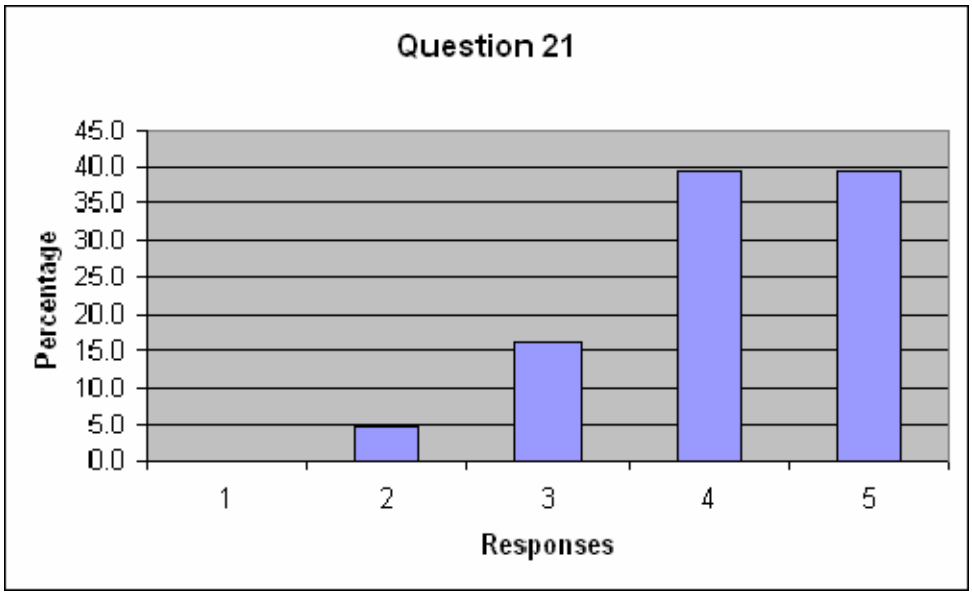
20. The use of monitoring tools will improve software quality.



Mean:	3.7442	Median:	4
Standard Deviation:	0.9535	Mode:	4
T-Value:	5.1179	P-Value:	<0.0001

Statistically Significant: Yes

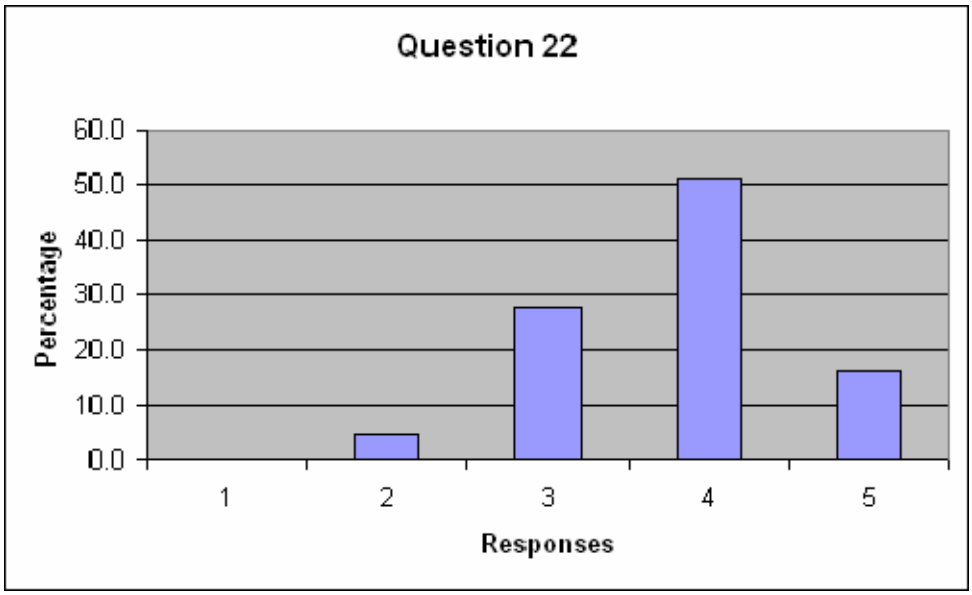
21. Service level agreements help manage customer's expectations.



Mean:	4.1395	Median:	4
Standard Deviation:	0.8614	Mode:	4
T-Value:	8.6750	P-Value:	<0.0001

Statistically Significant: Yes

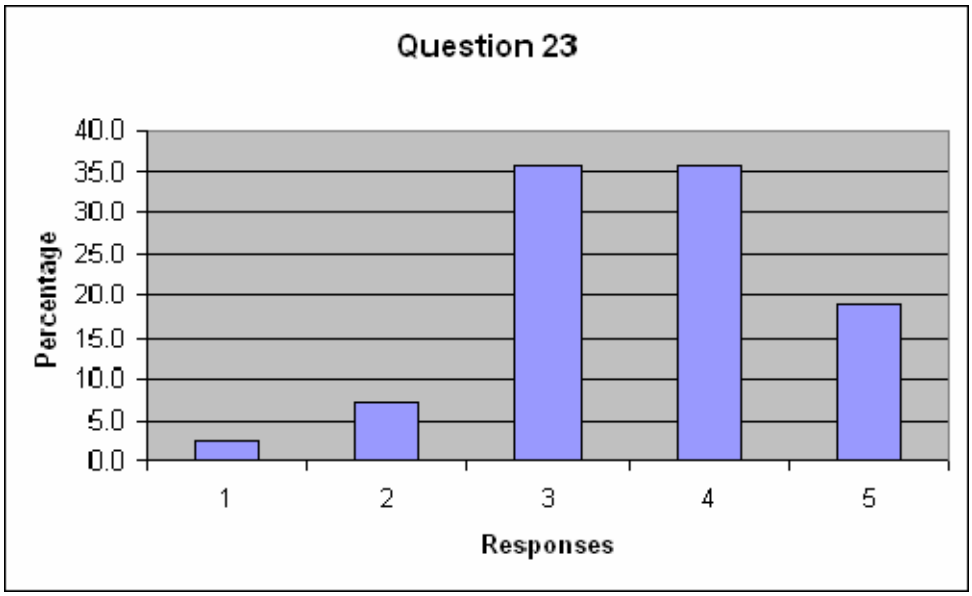
22. SLAs will help program managers focus on business critical measurements.



Mean:	3.7907	Median:	4
Standard Deviation:	0.7733	Mode:	4
T-Value:	6.7049	P-Value:	<0.0001

Statistically Significant: Yes

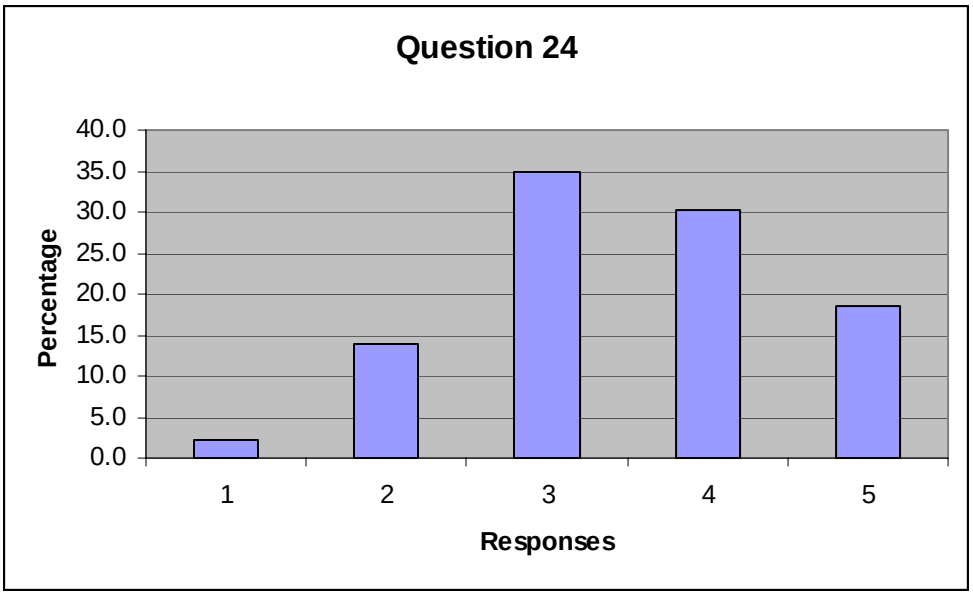
23. In a contract for application development, a SLA for availability will focus management attention on software maintainability, reliability and security early in the design stage, where it can be more easily implemented.



Mean:	3.6190	Median:	4
Standard Deviation:	0.9615	Mode:	3
T-Value:	4.1725	P-Value:	0.0002

Statistically Significant: Yes

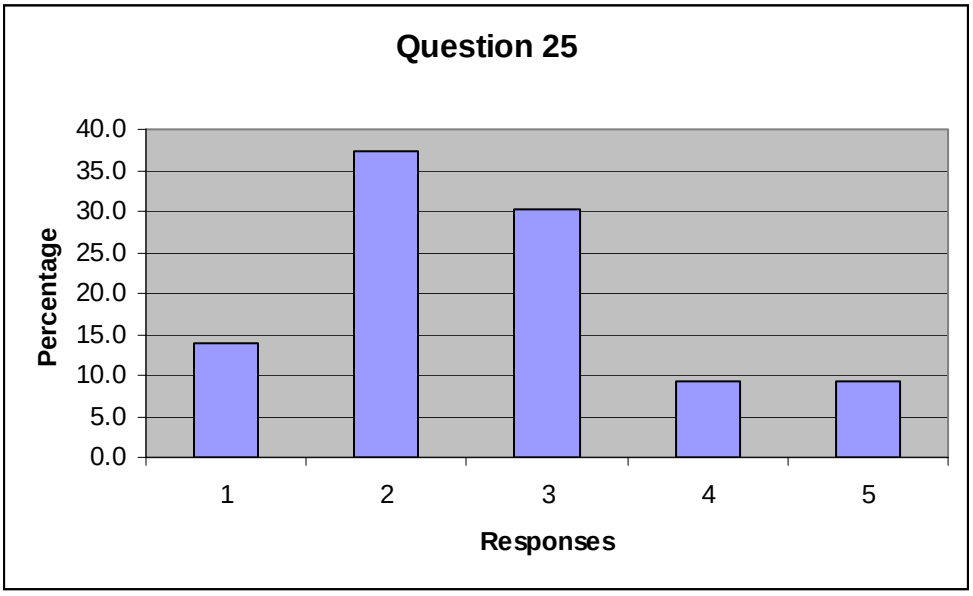
24. The SLAs would make source selection of potential service providers easier.



Mean:	3.4884	Median:	3
Standard Deviation:	1.0322	Mode:	3
T-Value:	3.1027	P-Value:	0.0034

Statistically Significant: Yes

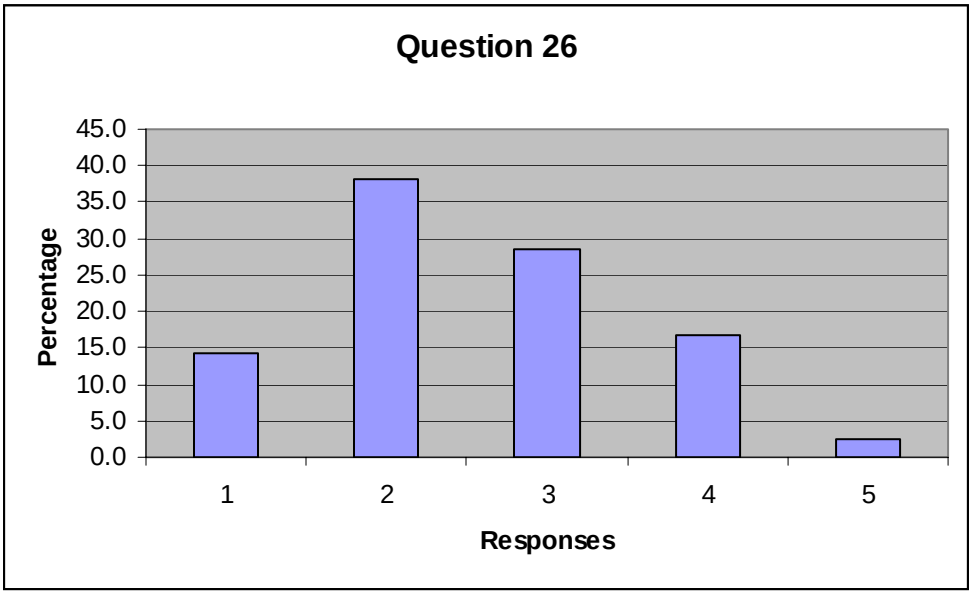
25. The administrative burden of managing the SLAs would outweigh their benefit.



Mean:	2.6279	Median:	2
Standard Deviation:	1.1344	Mode:	2
T-Value:	-2.1509	P-Value:	0.0373

Statistically Significant: Yes

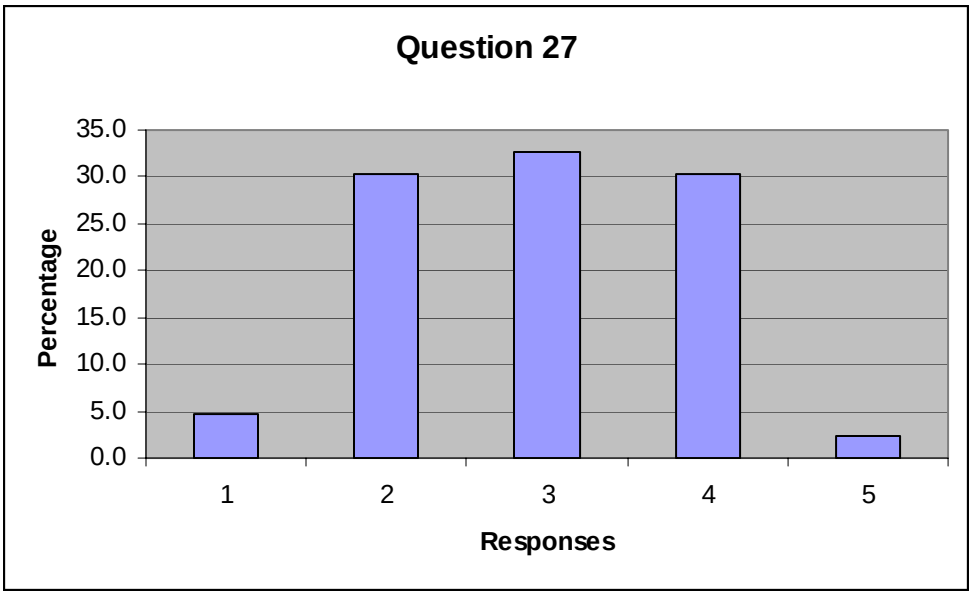
26. The difficulty in developing the SLAs would be too cumbersome for organizations.



Mean:	2.5476	Median:	2
Standard Deviation:	1.0170	Mode:	2
T-Value:	-2.8828	P-Value:	0.0062

Statistically Significant: Yes

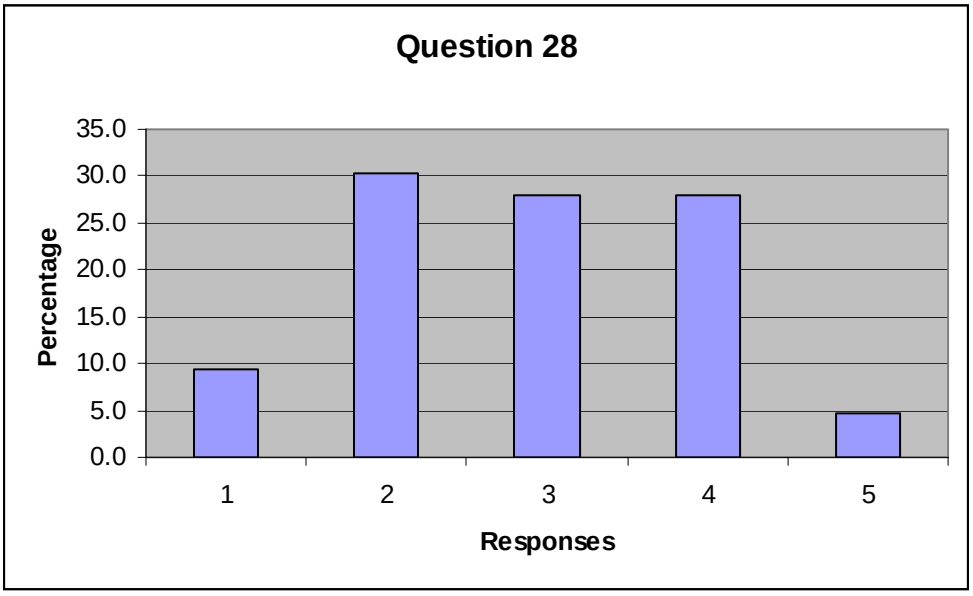
27. Service level agreements would not be developed because the people with the knowledge base to develop the SLAs were outsourced, or are not available.



Mean:	2.9535	Median:	3
Standard Deviation:	0.9500	Mode:	3
T-Value:	-0.3210	P-Value:	0.7498

Statistically Significant: No

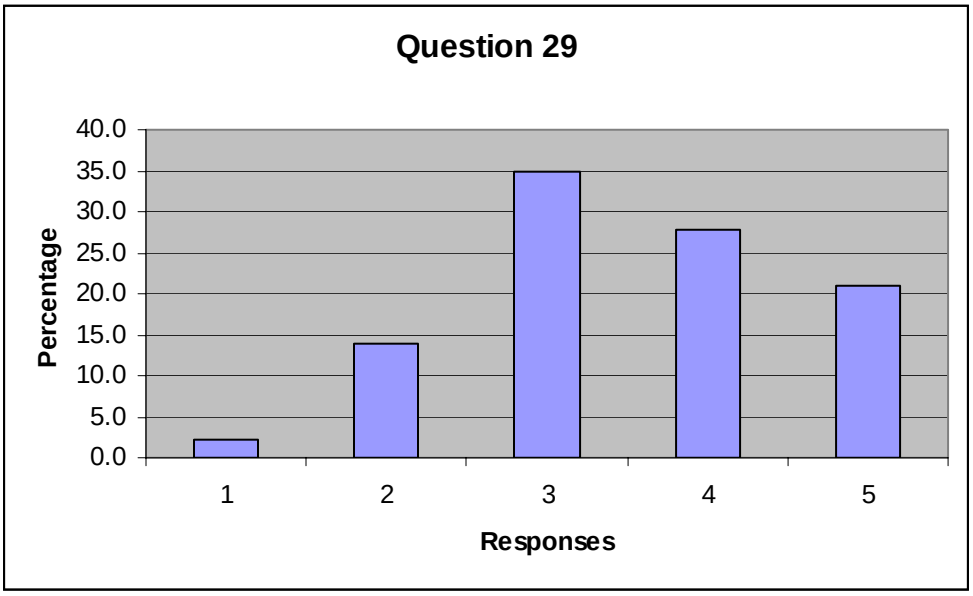
28. Enforcing penalty clauses/withholding incentives is too difficult.



Mean:	2.8837	Median:	3
Standard Deviation:	1.0737	Mode:	2
T-Value:	-0.7102	P-Value:	0.4815

Statistically Significant: No

29. Service level agreements will not resolve the quality issues associated with rushing software to market.



Mean:	3.5116	Median:	3
Standard Deviation:	1.0550	Mode:	3
T-Value:	3.1802	P-Value:	0.0028

Statistically Significant: Yes

30. Comments: If you have any comments, please group them into the following categories:

A. EFFECTIVENESS OF SLAS IN SOFTWARE ACQUISITION

- As you said in the introduction, the presence of standards for developing software does not mean they are being used. While, theoretically, a SLA would improve upon this, I believe we see too many instances today where SLAs are in use but fail to make this improvement, generally, in my opinion due to requirement changes. I do believe, however, that concerted use of SLAs should be a major aid in all the areas you address (contribution to software quality, post-production support, etc.)
- SLAs are only as effective as the people and organizations supporting them. Strong leadership, knowledge of the SW lifecycle and a good working relationship between the developer and the user can help to ensure quality. SLAs are really nothing more than contracts, and they are only as good as the enforcement.
- The flaw with the SLA used in the example is it only covers the host environment. The application is maintained by a separate organization and the network by a third

organization. If the SLA covered the entire suite, application, host, network and client, then one person would be responsible for service to the end user. With responsibility clearly in the hands of one organization, that organization can then be held accountable for the performance from the end users perspective.

- Good guide. Referential users better.
- Probably minimal.
- Most of the questions above are accurate only if the two parties involved, enforce the requirements identified in the SLA and the requirements are identified satisfactorily.
- A well-written RFP is appropriate before the SLA comes into play, but the SLA can be utilized to manage the on-going process in that it will force the contractor to hire more qualified programmers or make sure their technicians providing service are more qualified so that they can meet the SLA/RFP requirements. That will improve the entire cycle: requirements and demand.
- Software vendors are not held accountable to agreed upon capabilities. Procurement seems to lean toward buying new and not holding vendors accountable.
- I think that best use, as they will improve expectations.
- Good measure of forcing everyone to document expectations will help.
- Probably the standing.
- SLAs are only as good as it is managed.
- Not sure whether you are referring to custom development or off-the-shelf or both.
- For custom applications, it is exceptionally effective.
- In my opinion, effective metrics should be applied at the varying levels of a system (e.g. hardware, transport, OS, application, overall system).
- Software acquisition is always difficult and developing an SLA for software acquisition is much harder than for hardware and other more tangible items. A comprehensive SLA will never be able to capture all the minimum requirements needed and the problem that you face is the lack of flexibility in changing an SLA. At the various stages in the acquisition reviews of the SLA and how the development of the software is progressing should be conducted and the flexibility of adjusting the SLA should be available.

B. USEFULNESS OF THE SLA FORMAT

- The format must be simple, transparent and readable. The example given is clear and straightforward.
- Anyone with a IT background enjoys them.
- Format usually needs focusing and re-writing.
- Appropriate for the requirements.
- Any template or format will aid in a mutual focus on general topic areas. However, a mutual understanding of what each area really means would be key to its usefulness determination. It will only be as good as the data/words actually input for each area.

- Provides a structured framework to ensure all areas are discussed, managed and evaluated.
- Easy to understand.
- SLAs should be used more often.
- The format presented should be very useful in the creation of SLAs for operational support. I think SLAs for custom application development require a somewhat different SLA.
- Very useful.
- I like the clear definition of the SLA's intent; the method of calculation and definition of roles for the supplier and owner.

C. SLAS CONTRIBUTION TO SOFTWARE QUALITY

- SLAs can contribute to SW quality, but do not necessarily provide a silver bullet.
- Too soon to tell.
- If the company developing the applications is also responsible for the O&M of the application, then an SLA is a strong motivator for developing a quality system. How ever if the developer just hands the application over to the O&M provider, the incentive to the developer is removed.
- Can substantially improve quality from the get go.
- Microsoft themselves have proven that software only needs to be slightly better than what is out there. Quality software will always be rare despite SLAs.
- Great to prescribe document a set of criteria and/or data elements. However, quality goes back to the commitment and support by the team. Quality is not just setting a criteria/goal.
- Am doubtful that it will have a significant impact.
- Just OK, Not great or bad
- Again, if someone is actually monitoring, then SLAs will work.
- At face value an SLA could significantly improve software quality by providing stakeholders (including the CDA) a means to agree upon requirements and performance standards. However, complete and accurate requirements gathering is the key to software quality and having the ability to track software adherence to identified requirements throughout the development process and production upgrades (cradle to grave) in addition to the SLA would provide for the best quality in the delivered product.
- I think SLAs that mandate conformance to developmental standards and architectural guidelines can contribute immeasurably to software quality and usefulness over time. It is difficult to write an SLA that guarantees efficient coding.
- SLAs are a good tool to monitor operations and control expectations and such. I believe an SLA is an awkward tool to monitor software quality.
- A clear objective from the beginning will increase the probability of success for many application software-development activities. However, a changing market, compressed delivery needs and requirements evolving late in the development cycle will generally force decisions to move forward at the expense of quality. Furthermore, the cost of not having software may be greater than benefit of waiting until bugs are resolved.

- SLA could contribute to SQA but it is not the key contributor.
- I don't buy the basic thesis that SLAs would improve software quality. There is too much separation between the worlds. The software developers know nothing about, and never think about, the server/hosting environment that their applications will run on. It's their job to write the code and "throw it over the wall" to the network administrators. Only one of the items in "why software is so bad" is related to running the application once it is programmed. If you try to hold programmer's feet to the fire re: a SLA for availability, they are only going to point fingers at the network administrators and say, "it's their problem". The only person who cares about both worlds is the contract or program manager.

D. SLAS CONTRIBUTION TO POST-PRODUCTION SUPPORT

- SLAs can help to ensure that the proper post-production support is put into place, but they will not ensure success.
- Critical. Must have.
- Contribution will be dependent on the support being provided. For example, maintenance of a program would depend on the complexity of program, the type of problem reported, the recommended solution to fix the problem, etc.
- Probably none.
- Better then average.
- If post-production means maintenance and modification/enhancement, it is important to have an agreement that continues to ensure software quality as above.
- Consider the following items in an SLA for monitoring production. Blackout periods for servers during backup periods as your CPU utilization will probably be 100 percent for a significant period of time. This is a typical scenario that should not trigger a review action. Another item would be system availability. Never use ping to see if a server is available. Always use telnet. You can have a box in single-user mode that ping will detect but that server is not available for applications.
- Ensures support from multiple ends, from software design to IT implementation.
- SLAs definitely aid in the effective management of systems after delivery to production. Too often the end of development is considered the end of system investment.
- Most of the things that cause the application to go down are not application issues; their operating system and platform issues. Windows 2000 is better than prior versions, but I still consider it a 6 x 24 operating system in a 7 x 24 world. The things that cause our system to go offline are: server OS crashes, applying security patches that require servers to be rebooted, simple configuration tweaks that require rebooting etc… If the damn OS would stay running, then our applications would only be offline for scheduled maintenance. UNIX systems are totally different: we ran Oracle on a Sun box for 5 years with no reboots.

E. SLAS CONTRIBUTION TO LIFECYCLE MANAGEMENT

- Big benefit here. SLAs can help to define what the level and requirements for post-production support are intended to be.
- Serious. Need to have.
- It is a starting point for the project and can be used to verify successfully meeting customer documented expectations.
- Probably none.
- Just Ok, More monitoring and testing needs to be done.
- Should be a staff that does exclusive SLA management. SLAs are written but no one seems to really monitor them. There is still "no consequence for actions".
- SLAs theoretically could be used for configuration management and lifecycle maintenance assuming all parties involved maintain adherence to the SLA (to include change management of the SLA itself).
- This gets really hard over time, particularly when software acquisition, and operational support responsibilities are divided between multiple vendors and government agencies, as in the case study. Not impossible, but hard to keep multiple SLAs synchronized.
- SLAs contribute to the program lifecycle and should benefit development by having clearer requirements earlier in the cycle. Effectiveness of their contribution, however, is typically a derivative of what the author knew at the time of their creation. If a clear vision was available, SLA's will be effective. If the vision is vague and constantly evolving, their contribution will be less effective.
- It could be very helpful if it integrated in an automated software process environment. If not, it could be viewed as extra work only.

LIST OF REFERENCES

Abdel-Hamid, T., and Madnick, S., "Lessons Learned From Modeling the Dynamics of Software Development," Communications of the ACM, Vol. 32, No. 12, Dec. 1989, pp.14-26.

Agarwal, A., How to Optimize Outsourcing Relationships. In Info-Tech Research Group (Ed.), *Info-Tech White Papers 2003*, Info-Tech, London Ontario, 2003.

Albin, S., *The Art of Software Architecture: Design Methods and Techniques*, John Wiley and Sons, Indianapolis, 2003.

Albrecht, A., "Measuring Application Development Productivity," in *GUIDESHARE: Proceedings of the IBM Application Development Symposium*, (Monterey CA. 1979), pp. 83-92.

Alexander, L., and Davis, A., "Criteria for Selecting Software Process Models," in *Proceedings of the Fifteenth Annual International Conference on Computer Software and Applications*, IEEE (Tokyo, Japan, 11-13 September 1991), pp. 521-528

Anthes, G., "Quality?! What 's That?," *Computerworld*, Oct. 13, 1997, pp. 75-76.

Archer, B., "The Management of Client-Server Systems," *ICL Systems Journal*, Vol. 9 Iss. 1, May 1994.

Aries, J., Banerjee, S., Brittan, M., Dillon, E., Kowalik, J., and Lixvar, J., "Capacity and Performance Analysis of Distributed Enterprise Systems," *Communications of the ACM*, Vol. 45, Issue 6, pp. 100-105, Jun. 2002.

Armour, P., *The Laws of Software Process: A New Model for the Production and Management of Software*, Auerbach Publications, Boca Raton, 2004.

Atre, S., "The Hidden Costs of Client/Server," *DBMS*, Jun. 1995.

Aweya, J., Ouellette, M., Montuno, D., Doray, B., and Felske, K., "An Adaptive Load Balancing Scheme for Web Servers," *International Journal of Network Management*, Vol. 12, 2002, pp. 3-39.

Aweya, J., Ouellette, M., Montuno, D., Doray, B., and Felske, K., "An Adaptive Load Balancing Scheme for Web Servers," International Journal of Network Management, Vol. 12, No. 1, Jan./Feb. 2002, pp. 3-39.

Baker, M., "Implementing An Initial Software Metrics Program," in *Proceedings of the IEEE Aerospace and Electronics Conference*, (Atlanta, 20 May 1991), pp. 1289-1294.

Banker, R., et. al., "Software Complexity and Maintenance Costs" *Communications of the ACM*, Vol 36. No. 11, Nov. 1993, pp. 347-358.

Baron, R., and Williams, F., Chapter 13: Smart Outsourcing – The Legalities; How to Negotiate the Best Contract. In Reuvid, J., and Hinks, J. (Ed.), *Managing Business Support Services: Strategies for Outsourcing and Facilities Management, Second Edition*, Kogan Page, London, 2001.

Basil, V., Briand, L., and Melo, W., "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering*, Vol. 22, No. 1, Oct. 1996, pp. 751-761.

Bass, L., Clements, P., and Kazman, R., *Software Architecture in Practice*, Addison-Wesley, Reading, 1998.

Beath, C., and Walker, G., "Outsourcing of Application Software: A Knowledge Management Perspective," in *Proceedings of the Thirty First Hawaii International Conference on System Sciences*, IEEE (Kohala Coast, Hawaii, Jan. 1998) Vol. 6, pp. 666-674.

Becker, S., and Jorgensen, A., Chapter 17: A Recursive Approach to Software Development. In Valenti, S. (Ed.), *Successful Software Reengineering*, Idea Group Publishing, Hershey, 2002.

Bennatan, E., *On Time Within Budget*, Wiley Computer Publishing, New York, 2000.

Bennett, K., and Rajlich, V., "Software Maintenance and Evolution: A Roadmap," in *Proceedings of the Conference on the Future of Software Engineering*, ACM (Limerick, Ireland, Jun. 2000), 73-87.

Bertoa, M., and Vallecillo, A., "Quality Attributes for COTS Components," [http://www.sd-cenidet.com.mx/Revista/Docs/Vol1/No2Art06.pdf], Nov. 2002. Accessed Jul. 2004.

Berzins, V., and Luqi, *Software Engineering with Abstractions*, Addison-Wesley Publishing, Reading, 1991.

Bickerton, M., and Siddiqi, J., "The Classification of Requirements Engineering Methods," in *Proceedings of IEEE International Symposium on Requirements Engineering*, IEEE (San Diego, Jan. 1993), pp. 182-186.

Blacharski, D., "Keeping the Bugs at Bay," *IT World.Com*, 23 July 2002.

Boehm, B., "A Spiral Model of Software Development and Enhancement," *Computer*, Vol. 21, No. 5, May 1988, pp. 61-72.

Boehm, B., et.al., *Software Cost Estimation with COCOMO II*, Prentice Hall, Upper Saddle River, 2000.

Boehm, B., "Software Risk Management: Principles and Practice," *IEEE Software*, Vol. 8 No. 1, Jan. 1991, pp. 32-41.

Boehm, B., Brown, J., and Lipow, M., "Quantitative Evaluation of Software Quality," in *Proceedings of the 2nd International Conference on Software Engineering*, ACM (San Francisco, CA. 1976), 592-605.

Briand, L., Melo, W., Seaman, C., and Basili, V., Characterizing and assessing a large-scale maintenance organization, in *Proceedings of the Seventeenth International Conference on Software Engineering*, ACM (Seattle, Wash., Apr. 1995), 133-143.

Briones, J. Chapter 31: Quality Information Services. In Brown, C. (Ed.) with Topi, H., *IS Management Handbook, 7th ed*., CRC Press LLC Auerbach, Boca Raton, 2000.

Bubenko, J, "Challenges in Requirements Engineering," in *Proceedings of the Second IEEE Symposium on Requirements Engineering*, IEEE (York England, Mar. 1995), pp. 160-162.

Buco, M., Chang, R., Laun, L., Ward, C., Wolf, J., and Yu, P., "Managing eBusiness on Demand SLA Contracts in Business Terms Using the Cross-SLA Execution Manager SAM," in *Proceedings of the Sixth International Symposium on Autonomous Decentralized Systems*, IEEE, (Pisa, Italy, 9 Apr. 2003), pp. 157-164.

Byron, D., "Understanding the Costs of Client/Server Computing," [http://c14055.nkfust.edu.tw/datapro/33574-1.htm.], Feb. 1996. Accessed Jan. 2002.

Calliss, D., and Callis, F., "Criteria for Selecting a Family of Software Indicators," in *Proceedings of the Seventeenth Annual International Conference on Computer Software and Applications*, IEEE, (Phoenix, Nov. 1993), pp. 408-413.

Charette, R., "Are We Developers Liars or Just Fools," *IEEE Software*, Vol. 12, No. 4, Jul. 1995, pp. 90-92.

Charette, R., "Large-Scale Project Management is Risk Management," *IEEE Software*, Vol. 13, No. 4, Jul. 1996, pp. 110-117.

Charette, R., "Managing Risk in Software Maintenance," *IEEE Software*, Vol. 14, No. 3, May/Jun. 1997, pp. 43-50.

Charvat, J., *Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects*, John Wiley and Sons, New York, 2003.

Chief of Naval Operations, “Naval Message 251530Z Jul 01,” Jul. 25, 2001.

Chorafas, D., *Integrating ERP, CRM, Supply Chain Management, and Smart Materials*, Auerbach, Baca Raton, 2001.

Chulani, S., Boehm, B., and Abts, C., “Software Development Cost Estimation Approaches – A Survey,” [sunset.usc.edu/publications/TECHRPTS/2000/ usccse2000-505/usccse2000-505.pdf], 1998. Accessed Jul 2004.

Chutchian-Ferranti, J., “Activity Based Costing,” *Computerworld*, Aug. 09, 1999.

Clark, D., and Fang, W., “Explicit Allocation of Best-Effort Packet Delivery Service,” *IEEE/ACM Transactions on Networking*, Vol. 6, No. 4, Aug. 1998.

Clarke, S., and Lehaney, B. Chapter 140: Human-Centred Methods in Information Systems: Boundary Setting and Methodological Choice. In M. Khosrowpour (Ed.), *Challenges of Information Technology Management in the 21st Century: 2000 Information Resources Management Association International Conference* Hershey: Idea Group Publishing. 2000

Clements, P., Bass, L., Kazman, R., and Abowd, G., “Predicting Software Quality by Architecture Evaluation,” in *Proceedings of the Fifth International Conference on Software Quality*, (Austin, Texas, Oct. 1995).

Clinton, W., “Executive Order 13011 of Jul. 16, 1996,” *Federal Register*, Vol. 61, No. 140, 19 Jul. 1996.

Conxion Corp., “10 Critical Issues to Consider if You’re Concerned About Your Hosting Service,” [http://www.conxion.net], 2001. Accessed May 2003

Cook, M., *Building Enterprise Information Architectures Reengineering Information Systems*, Prentice Hall, Upper Saddle River, 1996.

Coppick, C., and Cheatham, T., “Software Metrics for Object-Oriented Systems,” in *Proceedings of the 1992 ACM Annual Conference on Communications*, ACM (Kansas City, MI. 1992), pp. 317-322.

Cross, S., “A Quality Doctrine for Software: Do it Right the First Time,” in *Proceedings of the Ninth Asia-Pacific Software Engineering Conference*, IEEE (Gold Coast, Australia, 2002), pp. 187-194.

Cross, S., and Estrada, R., “DART: An Example of Accelerated Evolutionary Development,” in *Proceedings of the Fifth International Conference on Rapid System Prototyping*, IEEE (Grenoble, France, 1994), 177-183.

Czegel, B., *Running an Effective Help Desk*, Wiley, John & Sons, New York, 1998.

Dalal, A., and Jordan, S., "Improving User-Perceived Performance at a World Wide Web Server, in *Proceedings of Globecom 2001*, IEEE (San Antonio, TX., Nov. 2001), pp. 2465-2469.

Daniels, A., *Bringing Out the Best in People: How to Apply the Astonishing Power of Positive Reinforcement*, McGraw-Hill, New York, 2000.

Dart, S., *Configuration Management: The Missing Link in Web Engineering*, Artech House, Norwood, 2000.

Daskalantonakis, M., "Achieving Higher SEI Levels," *IEEE Software*, Vol. 11, No., 4, Jul. 1994, pp. 17-24.

Daskalantonakis, M., "A Practical View of Software Measurement and Implementation Experiences within Motorola," *IEEE Transactions on Software Engineering*, Vol. 18, No. 19, Nov. 1992, pp. 998-1010.

Davis, A., et al., "Identifying and Measuring Quality in a Software Requirements Specification," in *Proceedings of the First International Symposium on Software Metrics*, IEEE (San Diego, Jan. 1993), pp. 141-152.

De Waal, A., *Quest for Balance: The Human Element in Performance Management Systems*, John Wiley & Sons, New York, 2002.

Defense Acquisition Workforce Improvement Act (DAWIA), Public Law 101-510, Chapter 87 of Title 10, U.S. Code, 1990.

Department of Defense, Office of the Inspector General, "Management of Information Technology Equipment, Office of the Secretary of Defense," D-2001-096, Apr. 9, 2001.

Department of Defense, Office of the Inspector General, "Semiannual Reports to Congress April 1, 2000 to September 30, 2000," [http://www.dodig.osd.mil/sar/index.html], 2000. Accessed Jul. 2004.

Department of Defense, Office of the Inspector General, "Summary of Audits of Acquisition of Information Technology," D-2000-162, Jul. 13, 2000.

Department of Defense, Office of the Inspector General, "Use of the DoD Joint Technical Architecture in the Acquisition Process," D-2001-121, May 14, 2001.

Department of Defense, Under Secretary of Defense (Acquisition, Technology and Logisitics), "Guidebook for Performanced-Based Services Acquisition (PBSA) in the

Department of Defense," 02 Jan 2001. [http://www.acq.osd.mil/dpap/Docs/pbsaguide010201.pdf], Accessed Sep 2004.

Department of the Navy, Chief Information Officer, *Department of the Navy Information Management/Information Technology Workforce Strategic Plan Fiscal Years 2001-2006*, Department of the Navy, Washington D.C., May 2001.

Deputy Assistant Secretary of Defense (Deputy Chief Information Officer), *Clinger Cohen Act of 1996 and Related Documents*, May 2000.

Devore, J., *Probability and Statistics for Engineering and the Sciences 4th ed*., Duxbury Press, Pacific Grove, 1995.

Dias, M., and Vieira, M., "Software Architecture Analysis Based on Statechart Semantics," in *Proceedings of the 10th International Workshop on Software Specification and Design*, IEEE, (San Diego, Nov. 2000), pp. 133-137.

Dikel, D., Kane, D., and Wilson, J., *Software Architecture Organizational Principles and Patterns*, Prentice Hall, Upper Saddle River, 2001.

Dietz, T., "Evaluating Software Suppliers as Part of IBM's Overall Software Acquisition Process," in *Proceedings of the Acquisition of Software-Intensive Systems Conference*, SEI, (Washington DC, 28 Jan. 2003).

Dobrica, L., and Niemela, E., "A Survey on Software Architecture Analysis Measures," *IEEE Transactions on Software Engineering*, Vol. 28, No. 7, Jul. 2002, pp. 638-653.

Domberger, S., *The Contacting Organization: A Strategic Guide to Outsourcing*, Oxford University Press, Oxford, 1998.

Drummond, J. "Specifying Quality of Service for Distributed Systems based upon Behavior Models," Doctoral Dissertation, Naval Postgraduate School, May 2002.

Duncan, N., Beyond opportunism: A resource-based view of outsourcing risk, *in Proceedings of the 31st Annual Hawaii International Conference on System Sciences* IEEE (Kohala Coast, Hawaii, Jan. 1998) Vol. 6, pp. 675-684.

Eager, D., Vernon, M., and Zahorjan, J., "Minimizing Bandwidth Requirements for On-Demand Data Delivery," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 5, September/October 2001.

Eisner, H., "Reengineering the Software Acquisition Process Using Developer Off-The-Shelf Systems (DOTSS)," in *Proceedings of the IEEE Systems, Man and Cybernetics*, Vol. 5, 1995, pp. 3971-3976.

El Emam, K., and Madhavji, N., “Measuring the Success of Requirements Engineering Processes,” in *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, IEEE (York England, Mar. 1995), pp. 204-211.

Electronic Data Systems, NMCI Information Strike Force, [www.isf-nmci.com]. Jul. 2004.

Eljabiri, O, and Deek, F., “Toward a Comprehensive Framework for Software Process Modeling Evolution,” in *Proceedings of ACS/IEEE International Conference on Computer Systems and Applications*, IEEE (Beirut, Lebanon, 25-29 June 2001), pp. 488-491.

Ellett, T., “Client/Server Accounting and Chargeback,” [http://www.charge-back.com /_ARTICLE/article2.htm]. Accessed Jul. 2004.

Estublier, J., Leblang, D., Clemm, G., Conradi, R, Tichy, W., Van der Hoek, A., and Wilborg-Weber, D., “Impact on the Research Community On The Field of Software Configuration Management,” *Software Engineering Notes*, Vol. 27, No. 5, Sep. 2002, pp. 31-39.

Estublier, J., “Software Configuration Management: A Roadmap,” in *Proceedings of the Conference on the Future of Software Engineering*, ACM, (Limerick, Ireland, 2000), pp. 279-289.

Ewusi-Mensah, K., *Software Development Failures: Anatomy of Abandoned Projects*, The MIT Press, Cambridge, 2003.

Factor, A., *Analyzing Application Service Providers*, Sun Microsystems Press, Palo Alto, 2002.

Farbey, B., and Finkelstein, A., “Software Acquisition: A Business Strategy Analysis,” in *Proceedings of the fifth IEEE International Symposium on Requirements Engineering*, IEEE (Toronto, Canada, 2001), pp. 76-83

Feeny, D., and Willcocks, L. Chapter 18: Rethinking Capabilities and Skills in the Information Systems Function. In Currie, W. (Ed.) and Galliers, B. (Ed.), *Rethinking Management Information Systems*, Oxford University Press, Oxford, 1999.

Fenton, N., and Neil, M., “A Critique of Software Defect Prediction Models,” *IEEE Transactions of Software Engineering*, Vol. 25, No. 5, Sep./Oct. 1999, pp. 675-689.

Fenton, N, Krause, P., and Neil, M., “Software Measurement: Uncertainty and Causal Modeling,” *IEEE Software*, Vol. 19, No. 4, Jul./Aug. 2002, pp. 116-122.

Ferguson, P., Humphrey, W., Khajenoori, S., Macke, S., and Matvya, A., "Results of Applying the Personal Software Process," *Computer*, Vol. 30, No. 5, May 1997, pp. 24-31.

Fishman, D., "Application Availability: An Approach to Measurement," [http://www.Nextslm.org/fishman.html], 2000. Accessed Jul. 2004.

Florac, W., "Software Quality Measurement: A Framework for Counting Problems and Defects," *Technical Report CMU/SEI-92-TR-022*, Sep. 1992.

Frost, C., Allen, D., Porter, J., and Bloodworth, P., *Operational Risk and Resilience*, Butterworth-Heinemann, Oxford, 2001.

Gaines, L., and Michael, B., "Service Level Agreements as Vehicles for Managing Acquisition in Software-Intensive Systems," *Defense Acquisition Review Journal*, Vol. 11, No. 3, Dec. 2004 – Mar. 2005, pp. 284-303.

Galin, D., "Software Quality Metrics – From Theory to Implementation," *Software Quality Professional*, Vol. 5, No. 3, Jun. 2003, pp. 24-31.

Galliers, R., and Swan, J., "Against structured approaches: Information requirements analysis as a socially mediated process," in *Proceedings of the thirteenth Hawaii International Conference on System Sciences* (Wailea, Hawaii, Jan. 1997), Vol 3., 179-187.

Gama, G., Meira, W., Carvalho, M., Guedes, D., and Almeida, V., "Resource Placement in Distributed e-Commerce Servers," in *Proceedings of Globecom 2001*, IEEE (San Antonio, TX., Nov. 2001), pp. 1677-1682.

Garlan, D., "Software Architecture: A Roadmap," in *Proceedings of the conference on the Future of Software Engineering*, ACM (Limerick, Ireland, 2000) pp. 93-101.

Garlan, D., Allen, R., and Ockerbloom, J., "Architectural Mismatch: Why Reuse is so Hard," *IEEE Software*, Vol. 12, No. 6, Nov. 1995, pp. 17-26.

Garvin, D., "What Does 'Product Quality' Really Mean?," *Sloan Management Review*, Vol. 26, No. 1, Fall 1984, pp. 25-43.

Gates, B., *Business @ The Speed of Thought Using a Digital Nervous System*, Warner Books, New York, 1999.

Gemmer, A., "Risk Management: Moving Beyond Process**"**, *IEEE Computer*, May 1997

Gibson, R., “Software Process Modeling: Theory, Results and Commentary,” in *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, IEEE (Kohala Coast, Hawaii, 6-9 Jan 1998), pp. 399-408.

Gilb, T., “Risk Management: A Practical Toolkit for Identifying Analyzing and Coping with Project Risk,” [http:www.result-planning.com], Jul. 2002. Accessed Jul. 2004.

Gilliam, D., Wolfe, T., Sherif, J., and Bishop, M., “Software Security Checklist for the Software Life Cycle,” in *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE, (Linz Austria, Jun. 2003), pp. 243-248.

Glass, R., “Defining Quality Intuitively,” *IEEE Software*, May 1998, pp. 103-107.

Gnatz, M., Marscahll, F., Popp, G. Rausch, A., Schwerin, W., “The Living Software Development Process,” *Software Quality Professional*, Vol. 5, No. 3, Jun. 2003, pp. 4-16.

Goldenson, D., and Gibson, D., “Demonstrating the Impacts and Benefits of CMMI: An Update and Preliminary Results,” [http://www.sei.cmu.edu/pub/documents /03.reports/pdf/03sr009-revised.pdf], Oct. 2003. Accessed 17 Jul. 2004.

Goodyear, M., et. al., *Enterprise System Architecture*, CRC Press, Boca Raton, 2000.

Goth, G., “The Ins and Outs of IT Outsourcing,” *IT Pro*, Jan./Feb. 1999, pp. 11-14.

Greaver, M., *Strategic Outsourcing: A Structured Approach to Outsourcing Decisions and Initiatives*, AMACOM, New York, 1999.

Griffith, A., and Curle, D. “Shortage of Skilled Talent Ranks as the No. 1 Challenge for the Information Content Industry,” *SIIA Press Release*, 12 Jan. 2004.

Gupta, S., and Sinha, M., “Impact of Software Testability Considerations on Software Development Life Cycle,” in *Proceedings of the First International Conference on Software Testing, Reliability and Quality Assurance*, IEEE (New Dehli, India, 21 December 1994), pp. 105-110.

Hamlet, D., and Voas, J., “Faults on its Sleeve: Amplifying Software Reliability Testing,” in *Proceedings of the 1993 International Symposium on Software Testing and Analysis*, AMC (Cambridge MA. 1993), pp. 89-98.

Hansen, W., “A Generic Process and Terminology for Evaluating COTS Software,” [http://www.sei.cmu.edu/cbs/tools99/generic/generic.pdf], 07 Sep. 1999. Accessed Jul. 2004.

Harney, J., *Application Service Providers*, Addison-Wesley, Boston, 2002.

Harris, M., "How not to Conduct a Configuration Management Program," *Logistics Spectrum*, Vol. 31, No. 1, Jan./Feb. 1997, pp. 7-10.

Harter, D., and Slaughter, S., "Process Maturity and Software Quality: A Field Study," in *Proceedings of the Twenty First International Conference on Information Systems*, ACM (Brisbane, Australia, 2000), 407-411.

Heeks, R., Krishna, S., Nicholson, B., and Sahay, S., "Synching or Sinking: Global Software Outsourcing Relationships," *IEEE Software*, Mar./Apr 2001.

Heldman, W., *IT Project+ Study Guide*, Sybex, Alameda, 2002.

Herrin, W., "Software Maintenance Costs: A Quantitative Evaluation," in *Proceedings of the 16th SIGCSE technical symposium on computer science education*, ACM (New Orleans, Mar. 1985), pp. 233-237.

Hickey, A., Dean, D., and Nunamaker, J., "Establishing a Foundation for Collaborative Scenario Elicitation," *ACM SIGMIS Database*, Vol. 30 No. 3-4 Summer/Fall 1999.

Hilburn, T., Townhidnejad, M., "Software Quality: A Curriculum Postscript," in *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education*, ACM (Austin TX. 2000), pp. 167-171

Hiles, A., *E-Business Service Level Agreements: Strategies for Service Providers, E-Commerce and Outsourcing*, Rothstein Associates, Connecticut, 2002.

Hill, G., *The Complete Project Management Office Handbook*, Auerback, Boca Raton, 2004.

Hochstetler, S, et. al., *Tivoli Netview 6.01 and Friends*, IBM Redbook, Austin, 2000.

Horch, J., *Practical Guide to Software Quality Management*, Artech House Publishers, Boston, 1996.

Huckle, T., "Collection of Software Bugs," [http://wwwzenger.informtik.tu-muenchen.de /persons/huckle/bugse.html], 4 July 2002. Accessed Jan 2003.

Hulse, C., et. al., "Reducing Maintenance Costs Through the Application of Modern Software Architecture Principles," in *Proceedings of the SIGAda Annual International Conference*, ACM (Redondo Beach, Calif., Oct. 1999), 101-110.

Humphrey, W., "Using a Defined and Measured Personal Software Process," *IEEE Software*, May 1996, pp. 77-88.

Hunter, G., Chapter 16: Qualitative Research in Information Systems: An Exploration of Methods, In Whitman, M, and Woszczynski, A., (Ed.) *The Handbook of Information Systems Research*, Idea Group, Hershey, 2004.

Hurst, D., Operational Availability Modeling for Risk and Impact Analysis, in *Proceedings of Reliability and Maintainability Symposium*, IEEE (Washington D.C. Jan. 1995), pp. 391-396.

IEEE Std 1061-1998, "IEEE Standard for a Software Quality Metrics Methodology," Dec. 31, 1998.

IEEE Std 1062, "IEEE Recommended Practice for Software Acquisition," Dec. 22, 1998.

In, H., Boehm, B., Rodgers, T., and Deutsch, M., "Applying WinWin to Quality Requirements: A Case Study," in *Proceedings of the 23rd International Conference on Software Engineering*, ACM (Toronto, Ontario, Canada, 2001) pp. 555-564.

Info-Tech Research Group, *Technology and Organizational Performance*, Info-Tech Research Group, London, 2001.

Itbug, "Client/Server Benchmarking, PC KPIs and TCO," [http://www.itbug.ndirect.co.uk/services/tco.htm]. Accessed Jul 2004.

Itbug, "Sample Case Study Report For Client/Server Platforms," [http://www.itbug.ndirect.co.uk/samples/Caserep.htm]. Accessed Jul 2004.

Johnson, S., *Who Moved My Cheese?*, G. P. Putnam's Sons, New York, 1998.

Jones, C., "Determining Software Schedules," *Computer*, Vol. 28, No. 2, Feb. 1995, pp.73-75.

Jones, C., "What are Function Points," [http://www.spr.com/library/ofuncmet.htm], 1997. Accessed Feb. 2003.

Joodi, P., and Burklo, J., "DoD Software Acquisition Management Overview," [http://www.stsc.hill.af.mil/crosstalk/1997/04/dodacquisition.asp], Apr. 1997. Accessed Jul. 2004.

Kajko-Mattsson, M., "A Conceptual Model of Software Maintenance," in *Proceedings of the International Conference on Software Engineering*, ACM (Kyoto, Japan, Apr. 1998), 422-425.

Kaner, C., "The Impossibility of Complete Testing," [http:www.kaner.com/imposs.htm], Nov. 1997. Accessed July 2004.

Kabay, M., *The NCSA Guide to Enterprise Security Protecting Information Assets*, McGraw-Hill, New York, 1996.

Kafura, D., "A Survey of Software Metrics," in *Proceedings of the 1985 ACM Annual Conference on the Range of Computing: Mid-80's Perspective*, ACM (Denver, CO. 1985), pp. 502-506.

Kajko-Mattsson, M., "A Conceptual Model of Software Maintenance," in *Proceedings of the 20th International Conference on Software Engineering*, IEEE (Kyoto, Japan, 19 Apr. 1998), pp. 422-425.

King, W., "Guest Editorial Developing a Sourcing Strategy for IS: A Behavioral Decision Process and Framework," *IEEE Transactions on Engineering Management*, Vol. 48, No. 1, Feb. 2001, pp. 15-24.

Kirk, D., "A Flexible Software Process Model," in *Proceedings of the 26th International Conference on Software Engineering*, ACM (Edinburgh, Scotland, 23-28 May 2004) pp. 57-59.

Kitchenham, B., and Linkman, S., "Estimates, Uncertainty, and Risk," *IEEE Software*, Vol. 14, No. 3, May/June 1997, pp. 69-75.

Kitchenham, B., Pickard, L., and Linkman, S., "An Evaluation of Some Design Metrics," *Software Engineering Journal*, Vol. 5, No. 1, Jan. 1990, pp. 50-58.

Keil, M., Cule, P., Lyytinen, K., and Schmidt, R., "A Framework for Identifying Software Project Risk," *Communications of the ACM*, Vol. 41, No. 11, Nov. 1998.

Kendrick, T., *Identifying and Managing Project Risk: Essential Tools for Failure-Proofing Your Project*, AMACOM, New York, 2003.

Kern., T., and Willcocks, L., Chapter XV: Contract, Control and "Presentation" in IT Outsourcing – Research in Thirteen UK Organizations. In Tan., F. (Ed.), *Advanced Topics in Global Information Management*, Idea Group Publishing, Hershey, 2002.

King, W., "Guest Editorial Developing a Sourcing Strategy for IS: A Behavioral Decision Process and Framework," *IEEE Transactions on Engineering Management*, Vol. 48, No. 1, Feb. 2001.

Kobitzch, W., Rombach, D., and Feldman, R., "Outsourcing in India," *IEEE Software*, Mar./Apr. 2001.

Krogstie, J., "Using Quality Function Deployment in Software Requirement Specifications" [http://www.ifi.uib.no/konf/refsq99/papers/krogstie.rtf], Accessed 27 Sep 04.

Kumar, A., Kumar, P., and Basu, S. Chapter 10: Student Perceptions of Virtual Education: An Exploratory Study. In M. Khosrow-Pour (Ed.), *Web-Based Instructional Learning*, Idea Group Publishing, Hershey, 2000.

Kuver, P., Chapter 11: SEI CMM or ISO 9000: Which is Right For Your Organization in Tinnirello, P. (Ed.), *New Directions in Project Management*, Auerbach Publication, Boca Raton, 2002.

Lacity, M., and Hirschheim, R. Information technology outsourcing: What problems are we trying to solve? In *Rethinking Management Information Systems*, Oxford University Press, London, 1999.

Land, R., "Improving Quality Attributes of a Complex System Through Architectural Analysis – A Case Study," in *Proceedings of the Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, IEEE, (Lund Sweden, Apr. 2002), pp. 167-174.

Lazarescu, M., Bammi, J., Harcourt, E., Lavagno, L., and Lajolo, M., "Compilation-Based Software Performance Estimation for System Level Design," in *Proceedings of IEEE International Workshop on High-Level Design Validation and Test*, IEEE, (Berkeley, Nov. 2000), pp. 167-172.

Lee, J., and Ben-Natan, R., *Integrating Service Level Agreements – Optimizing your OSS for SLA Delivery*, Wiley Publishing, Indianapolis, 2002.

Leon, A., *A Guide to Software Configuration Management*, Artech House, Norwood, 2000.

Lewis, H., *Bids, Tenders and Proposals: Winning Business Through Best Practice*, Kogan Page, London, 2003.

Liebmann, L., "Lessons from the Toy Factory: Treat IT as a Raw Material," *Computerworld*, [http://www.computerworld.com/news/1999/story/0,11280,34787,00.html], Mar. 08, 1999. Accessed Jul. 2004.

Loeb, V., and Schneider, G., "NSA Picks Information Technology Contractor," *Washington Post*, Aug. 1, 2001.

Loosley, C., and Douglas, F., *High-Performance Client/Server*, John Wiley & Sons, New York, 1998.

MacCormack, A., Kemerer, C., Cusumano, M., and Crandall, B., "Trade-offs Between Productivity and Quality in Selecting Software Development Practices," *IEEE Software*, September/October 2003, pp. 78-85.

Machniak, M., "Development of a Quality Management Metric (QMM) Measuring Software Program Management Quality," Masters Thesis, Naval Postgraduate School, December 1999.

Malhotra, Y., "IS Productivity and Outsourcing Policy: A Conceptual Framework and Empirical Analysis," *in Proceedings of Inaugural Americas Conference on Information Systems*, pp.142-144, Aug. 25, 1995.

Mann, C., "Why Software is so Bad," *MIT's Magazine of Innovation Technology Review*, Aug. 2002.

Markset, T., and Kumar, U., "R&M and Risk Analysis Tools in Product Design to Reduce Life-Cycle Cost and Improve Attractiveness," in *Proceedings of the Annual Reliability and Maintainability Symposium*, IEEE, (Piscataway, NJ 2001), pp. 116-122.

Martin, R., and Raffo, D., "A Comparison of Software Process Modeling Techniques," in *Proceedings of the Portland International Conference on Management and Technology*, IEEE (Portland Oregon, 27 July 1997), pp. 577-580.

Mater, J., Chapter 16: Solving the Software Quality Management Problem. In Tinnirello, P. (Ed.), *New Directions in Project Management*, CRC Press LLC, Boca Raton, 2002.

Matloff, N., "Problems and Needed Reform for the H-1B and L-1 Work Visas," [http:heather.cs.ucdavis.edu/itaa.others.html], Feb. 9, 2004. Accessed Apr. 2004.

Mayrand, J., and Coallier, F., "System Acquisition Based on Software Product Assessment", *Proceedings of the 18th International Conference on Software Engineering*, 1996, pp. 210-219.

McClure, C., "A Model for Program Complexity Analysis," in *Proceedings of the 3rd International Conference on Software Engineering*, IEEE, (Los Alamitos, CA. May 1978) pp.149-157

McConnell, S., "Less is More," [http://www.stevemcconnell.com/articles/art06.htm], October 1997. Accessed Jul. 2004.

McConnell, S., Rapid Development: Taming Wild Software Schedules, Microsoft Press, Redman, WA., 1996.

McGuire, E., "Factors Affecting the Quality of Software Project Management: An Empirical Study Based on the Capability Maturity Model," *Software Quality Journal 5*, 1996, pp. 305-317.

McLaughlin, L., "An Eye On India: Outsourcing Debate Continues," *IEEE Software*, Vol. 20, No. 3, May-Jun. 2003.

Mei, H., Zhang, L., and Yang, F., "A Software Configuration Management Model for Supporting Component-Based Software Development," *Software Engineering Notes*, Vol. 26, No. 2, Mar. 2001, pp. 53-58.

Menasce, D., and Gomaa, H., "A Method for Design and Performance Modeling of Client/Server Systems," *IEEE Transactions on Software Engineering*, Vol. 26, No. 11, Nov. 2000, pp. 1066-1085.

Mengel, S., "Software Metrics: Views from Education, Research, and Training," in *Proceedings of the 12th Conference on Software Education and Training*, IEEE, (New Orleans, LA. 22-24 Mar. 1999.) pp. 1-3.

Microsoft Corporation, *Microsoft Windows 2000 Server Resource Kit: Microsoft Internet Information Services 5.0*, Microsoft Corporation, Redland, 2000.

Millard, E., "Probing the IT Skills Shortage," *E-Commerce Times*, Jul. 15, 2003.

Milosevic, Z., and Dromey, R., "On Expressing and Monitoring Behaviour in Contracts," in *Proceedings of the Sixth International Enterprise Distributed Object Computing Conference*, IEEE, (Lausanne, Switzerland, 17 Sep. 2002), pp. 3-14.

Minasi, M., Anderson, C., Smith, B., and Toombs, D., *Mastering Windows 2000 Server Third Edition*, Sybex, San Francisco, 2001.

Mingers, J. Chapter 204: Information Systems: The Case of the Missing Body. In M. Khosrowpour (Ed.), *Challenges of Information Technology Management in the 21st Century: 2000 Information Resources Management Association International Conference*, Idea Group Publishing, Hershey, 2000.

Montgomery, A., and Dolphin, L., "Is the Velocity of Light Constant in Time," *Galilean Electrodynamics* Vol. 4, No. 5, Sep./Oct. 1993.

Moody, D., "Building links between IS research and professional practice: improving the relevance and impact of IS research," in *Proceedings of the Twenty First International Conference on Information Systems* (Brisbane, Australia, 2000), 351-360.

Musa, J., "More Reliable Software Faster and Cheaper: An Overview of Software Reliability Engineering," [http://members.aol.com/JohnDMusa/ARTweb.htm], Mar. 28, 2002. Accessed Jul. 2004.

Mylopoulos, J., Chung, L., and Yu, E., "From Object-Oriented to Goal-Oriented Requirements Analysis," *Communications of the ACM*, Vol. 42, No. 1, Jan. 1999.

Nellore, R., "Validating Specifications: A Contract-Based Approach," *IEEE Transactions on Engineering Management*, Vol. 48, No. 4, Nov. 2001.

Nelson, P., et. al., "Two Dimensions of Software Acquisition," *Communications of the ACM*, Vol. 39, N0. 7, Jul. 1996.

Nemeth, E., et. al., *Unix System Administration Handbook*, 3rd ed., Prentice Hall PTR, Upper Saddle River, 2001.

Nielsen, S., *Performance Considerations for Domino Applications*, IBM Redbooks, Poughkeepsie, 2000.

Nogueira de Leon, J., "A Formal Model for Risk Assessment in Software Projects," doctoral dissertation, Naval Postgraduate School, Sep. 2000.

Norris, G., Hurley, J., Hartley, K., Dunleavy, J., and D., J., *ERP/E-Business Impact on Shared Services*, John Wiley and Sons, New York, 2000.

Nuseibeh, B., and Easterbrook, S., "Requirements Engineering: A Roadmap," in *Proceedings of the Conference on the Future of Software Engineering*, ACM (Limerick, Ireland, 2000), 35-46.

Nutt, G., "Software Engineering Process Model – A Case Study," in *Proceedings of Conference on Organizational Computing Systems*, ACM (Milpitas, CA, 1995), pp. 324-335.

Office of Government Commerce (OGC), *Best Practice for Service Delivery: IT Infrastructure Library (ITIL) The Key to Managing IT Services*, The Stationary Officer, London, 2001.

Ogasawara, H., Yamada, A. and Kojo, M., "Experiences of Software Quality Management Using Metrics Through the Life-Cycle," in *Proceedings of the 18th International Conference on Software Engineering*, ACM (Berlin, Germany, 1996), 179-188.

Osmundson, J., Michael, J., Machniak, M., and Grossman, M., "Quality Management Metrics for Software Development," *Information and Management*, 2003. pp. 799-812.

Packeteer, Inc., "Gaining Visibility Into Application Performance," [http://whitepapers.zdnet.co.uk/0,39025945,60027476p-39000526q,00.htm], 2001. Accessed Jul. 2004.

Packeteer, Inc., "Shaping Application Behavior," [http://www.kappanetworks.com/research/bin/ShapingApplicationBehavior.pdf], May 2002. Accessed Jul. 2004.

Padayachee, K., "An Interpretive Study of Software Risk Management Perspectives," in *Proceedings of the 2002 Annual Research Conference of the South African Institute of Computer Science and Information Technologists on Enablement Through Technology*, ACM (Port Elizabeth, South Africa, 2002), pp. 118-127.

Palmer, J., and Evans, R., "Software Risk Management: Requirements-Based Risk Metrics," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, IEEE (San Antonio, TX., Oct. 1994), pp. 836-841.

Patton, R., and Ogle, J., *Designing SQL Server 2000 Databases for .Net Enterprise Servers*, Syngress Publishing, Rockland Md., 2001.

Paulk, M., "Software Process Improvement Minitrack Introduction 29th Hawaii International Conference on Systems Science," in *Proceeding of the 29th Hawaii International Conference on Systems Science*, IEEE (Wailea, HI., 3 Jan. 1996), pp. 671-672.

Pearce, T., and Oman, P., "Maintainability Measurements on Industrial Source Code Maintenance Activity," in *Proceedings of the International Conference on Software Maintenance*, IEEE, (Opio, France, Oct. 1995), pp. 295-303.

Peckinpaugh, C., "Is Government 'Computer Chaos' Over?," *Federal Computer Week*, Oct. 18, 1999.

Peltier, T., *Information Security Risk Analysis*, Auerbach Publishing, Boca Raton, 2001.

Philcox, J., *Solaris System Management*, New Riders, Indianapolis, 2001.

Plunkett, P., "Performance-Based Management Eight Steps to Develop and Use Information Technology Performance Measures Effectively," [http://www.gsa.gov/gsa/cm_attachments/GSA_DOCUMENT/eight_steps_R2GX2-u_0Z5RDZ-i34K-pR.doc] Accessed 17 Jul. 2004.

Portera, "When Does Using a Hosted Application Make Sense," [http:www.portera.com]. Accessed Jan. 2001.

Pressman, R., *Software Engineering A Practitioner's Approach*, 5th ed., McGraw-Hill, New York, 2001.

Pritchett, W., "An Object-Oriented Metrics Suite for Ada 95," in *Proceedings of the 2001 Annual ACM SIGAda International Conference on Ada*," ACM (Bloomington, MN. 2001), pp. 117-126.

Prouten, K., "Software Understanding and Reengineering Overview," [http://sunset.usc.Edu/Activities/workshop5o.html]. Accessed Jul. 2004.

Putnam, L., and Myers, W., *Measure for Excellence*, Yourdon Press Computing Series, Englewood Cliffs, NJ., 1992.

Raffo, D., Harrison, W., and Vandeville, J., "Software Process Decision Support: Making Process Tradeoffs Using a Hybrid Metrics, Modeling and Utility Framework," in *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, ACM (Ischia, Italy, 2002) pp. 803-809.

Rao, H., Nam, K., and Chaudhury, A., "Information Systems Outsourcing," *Communications of the ACM*, Vol. 39 No. 7, Jul. 1996.

Rakitin, S., *Software Verification and Validation: A Practitioner's Guide*, Artech House, Norwood, MA., 1997.

Reeves, T., "Educational Paradigms," [http:www.educationau.edu.au/archives/cp/REFS/Reeves_paradigms.htm], 21 Feb. 1996. Accessed Jul. 2004.

Reilly, G., "Server Performance and Scalability Killers," [http://msdn.Microsoft.com/library/en-us/dniis/html/tencom.asp] 22 Feb. 1999. Accessed Jul. 2004.

Richman, L., *Project Management Step-by-Step*, AMACOM, New York, 2002.

Rico, D., *ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers*, J. Ross, Boca Raton, 2004.

Riedel, T., Kaminski, J., Wahl, M, and Ambler, T., "Effective Testability Design for the Product Life-Cycle," in *Proceedings of the IEEE System Readiness Technology Conference*, IEEE, (San Antonio, August 1999), pp. 607-612.

Ripin, K., and Sayles, L., *Insider Strategies for Outsourcing Information Systems*, Oxford University Press, Oxford, 1999.

Robert, P., "Quality Requirements for Software Acquisition," in *Proceedings of the Third International Software Engineering Standards Symposium and Forum*, IEEE (Walnut Creek, CA, 1997) pp. 136-143.

Roberts, P., Chapter 12: Procurement – Best Value Criteria for Selection. In Reuvid, J., and Hinks, J. (Ed.), *Managing Business Support Services: Strategies for Outsourcing and Facilities Management, Second Edition*, Kogan Page, London, 2001.

Rocheleau, B., "Governmental Information System Problems and Failures: A Preliminary Review," [http:www.psdc.hbg.psu.edu/Faculty/jxr11/roche.html], 1997. Accessed Jan. 2002.

Royce, W., "Managing the Development of Large Software Systems: Concepts and Techniques," *Proceedings of WESCON*, August 1970, reprinted in *Proceedings of Ninth International Conference on Software Engineering*, IEEE (Monterey, CA, 1987), pp. 328-338.

Royce, W., “Pragmatic Quality Metrics for Evolutionary Software Development,” in *Proceedings of the Conference on TRI-ADA ’90*, ACM (Baltimore, MD. 1990) pp. 551-565.

Rutherford, E., “Hot Market, Cold Facts,” [http://webbusiness.cio.com/archive /012400_asp.html], Jan. 24, 2000. Accessed Jul. 2004.

Sarker, S., and Lee, A., “Using a Positivist Case Research Methodology to Test a Theory About IT-Enabled Business Process Redesign,” in *Proceedings of the International Conference on Information Systems* (Helsinki, Finland, 1998), 237-252.

Sarma, A., Noroozi, A., Van de Hoek, A., “Palantir: Raising Awareness Among Configuration Management Workspaces,” in *Proceedings of the 25th International Conference on Software Engineering,* ACM, (Portland, 2003), pp. 444-454.

Sawyer, P. and Kotonya, G., *Stoneman Version 2.0*, [http://www.swebok.org], May 2001. Accessed May 2003.

Scacchi, W., “Process Models in Software Engineering,” http://www.ics.uci.edu/ ~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf., October 2001. Accessed August 22, 2004.

Schmidt, M., *Implementing the IEEE Software Engineering Standards,* Sams, Indianapolis, 2000.

Schmietendorf, A., Dimitrov, E., and Dumke, R., “Process Models for the Software Development and Performance Engineering Tasks,” in *Proceedings of the Third International Workshop on Software and Performance,* ACM (Rome, Italy, 24-26 July 2002), pp. 211-218.

Schneidewind, N., “Body of Knowledge for Software Quality Measurement,” *IEEE Computer*, Feb. 2002, pp. 77-83.

Schneidewind, N., “Investigation of the Risk to Software Reliability and Maintainability of Requirements Changes,” in *Proceedings of the International Conference on Software Maintenance*, IEEE, (Florence IT. 6-10 Nov. 2001), pp. 127-136.

Schneidewind, N., “Methods for Assessing COTS Reliability, Maintainability, and Availability,” in *Proceedings of the Fifth International Conference on Software Maintenance*, IEEE (Bethesda, MD., Nov. 1998), pp. 224-225.

Schneidewind, N., “Software Metrics Model for Quality Control,” in *Proceedings of the Fourth International Software Metrics Symposium,* IEEE (Albuquerque, NM., 5 Nov. 1997), pp. 127-136.

Secretary of Defense, “Memorandum U09344/97,” Jun. 2, 1997.

Shaw, M., Garlan, D., *Software Architecture Perspectives on an Emerging Discipline*, Prentice Hall, Upper Saddle River, 1996.

Shekaran, C., Garlan, D, Jackson, M., Mead, N., Potts, C., and Reubenstein, H., “The Role of Software Architecture in Requirements Engineering,” in *Proceedings of the International Conference on Requirements Engineering*, IEEE (Colorado Springs, CO., 1994), pp. 239-245.

Shepperd, M., “Design Metrics: An Empirical Analysis,” *Software Engineering Journal*, Vol. 5, No. 1, Jan. 1990, pp. 3-10.

Simitci, H., *Storage Network Performance Analysis*, John Wiley & Sons, New York, 2003.

Simpson, J., Field, L., and Garvin, D., *The Boeing 767: From Concept to Production (A)*, 9-688-040, Harvard Business School, Cambridge, Apr. 1, 1991.

Sjouwerman, S., Shilmover, B., and Stewart, J., *Windows 2000 System Administrator’s Black Book*, Coriolis, Scottsdale, 2000.

Slabodkin, G., “Software Gliches leave Navy Smart Ship Dead in the Water,” *Government News*, Jul. 13, 1998.

Slaughter, S., Harter, D., and Krishnan, M., “Evaluating the Cost of Software Quality,” *Communications of the ACM*, Vol. 41, No. 8, Aug. 1998.

Smith, C., “Designing High-Performance Distributed Applications Using Software Performance Engineering: A Tutorial,” in *Proceedings of the Computer Measurement Group*, (San Diego, Dec. 1996), pp. 1-11.

Smith, C., and Williams, L., “Software Performance Engineering for Object-Oriented Systems: A Use Case Approach,” [http:www.perfeng.com/papers/uspood.pdf], 1998. Accessed Jul. 2004.

Smith, D., and Fletcher, J., *Inside Information: Making sense of Marketing Data*, John Wiley and Sons, New York, 2001.

Smith, M. A., Mitra, S, Narasimhan, S. “Offshore Outsourcing of Software Development and Maintenance: A Framework for Issues,” *Information and Management*, Vol 31, No. 3, Dec. 1996, pp. 165-175.

Smith, P., and Merritt, G., *Proactive Risk Management*, Productivity Press, New York, 2002.

Software Engineering Institute, Carnegie Mellon University, "Software Acquisition Capability Maturity Model (SA-CMM), Version 1.03" [http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr010.pdf], Mar. 2002. Accessed Jul. 2004.

Software Engineering Institute, Carnegie Mellon University, "Capability Maturity Model Integration (CMMI[SM]), Version 1.1," Aug. 2002.

Sommerville, I., Rodden, T., Sawyer, P., Bentley, R., and Twidale, M., "Integrating Ethnography into the Requirements Engineering Process," in *Proceedings of IEEE International Symposium on Requirements Engineering*, IEEE, (San Diego, Jan. 1993), pp. 165-173.

Sommerville, I., Sawyer, P., and Viller, S., "Viewpoints for Requirements Elicitation: A Practical Approach," in *Proceedings of the Third International Conference on Requirements Engineering*, IEEE (Colorado Springs, Apr. 1998), pp. 74-81.

Sopko, S., "Speeding up Service Level Agreement Negotiations," [http://www.nextslm.org/sopko1.html], 2002. Accessed Jul. 2004.

Sturm, R., Morris, W., and Jander, M., *Foundations of Service Level Management*, Sams, Indianapolis, 2000.

Sturm, R., *Working with Unicenter TNG*, Que, Indianapolis, 1998.

Sumner, M., "Risk Factors in Enterprise Wide Information Management System Projects," in *Proceedings of the 2000 ACM SIGCPR Conference on Computer Personnel Research*, ACM (Chicago, IL. 2000), pp. 180-187.

Surmacz, J., "Service with a Smile," [http://http://www.cio.com/outsourcing/edit/112900_service.html], Nov. 29, 2000. Accessed Jul. 2004.

Susarla, A., Barua, A., and Whinston, A., "Myths about Outsourcing to Application Service Providers," *IT Professional*, Vol. 3, No. 3, May-Jun. 2001.

Susarla, A., Barua, A., and Whinston, A., "Making the Most Out of an ASP Relationship," *IT Professional*, Vol. 3, No. 6, Nov.-Dec. 2001.

Sutcliffe, A., "Requirements Rationales: Integrating Approaches to Requirement Analysis," in *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods and Techniques*, ACM (Ann Arbor, 1995), pp. 33-42.

Suzuki, K., and Sangiovani-Vinventelli, "Efficient Software Performance Estimation Methods for Hardware/Software Codesign," in *Proceedings of the 33rd Automation Conference*, IEEE, (Las Vegas, Jun. 1996), pp. 605-610.

Tanenbaum, A., *Computer Networks Third Edition*, Prentice Hall, Upper Saddle River, 1996.

Tao, L., "Shifting Paradigms with the Application Service Provider Model," *Computer*, Vol. 34 No. 10, October 2001.

Tice, G., "Perspectives on Software Quality Assurance," in *Proceedings addendum of the 1985 ACM Annual Conference on the Range of Computing: Mid-80s Perspective*, ACM (Denver CO. 1985), p. 20.

Tricker, R., and Sherring-Lucas, B., *ISO 9001:2000 In Brief*, Butterworth Heinemann, Oxford, 2001.

Trochim, W. "What is Research Methods Knowledge Base?," [http://www.socialresearchmethods.net/kb/index.htm], 2002. Accessed Jul. 2004.

Turk, D., and Vaishnavi, V., Chapter 20: Software Process Models are Software Too: A Domain Class Model for Software Process Models. In Valenti S. (Ed.), *Successful Software Reengineering*, Idea Group, Hershey, 2002.

U.S. Air Force, "Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems, Command and Control, Management Information Systems Version 4.0," [http://www.stsc.hill.af.mil/resources/tech%5Fdocs/gsam4.html] Feb 2003. Accessed Jul. 2004.

U.S. Air Force, "Software Risk Abatement," *AFCS/AFLC Pamphlet 800-45*, 30 Sep. 1988.

U.S. General Accounting Office, "Defense Information Superiority: Progress Made, but Significant Challenges Remain," GAO/NSIAD/AIMD-98-257, Aug. 1998.

U.S. General Accounting Office, "Critical Infrastructure Protection: Challenges to Building a Comprehensive Strategy for Information Sharing and Coordination," GAO/T-AIMD-00-268, Jul. 26, 2000.

U.S. General Accounting Office, "Defense IRM: Alternatives Should be Considered in Developing the New Civilian Personnel System," GAO/AIMD-99-20, Jan. 27, 1999.

U.S. General Accounting Office, "Defense IRM: Critical Risks Facing New Material Management Strategy," GAO/AIMD-96-109, Sep. 06, 1996.

U.S. General Accounting Office, "Defense IRM: Strategy Needed for Logistics Information Technology Improvement Efforts," GAO/AIMD-97-6, Nov. 14, 1996.

U.S. General Accounting Office, “Defense Software: Review of Defense Report on Software Development Best Practices,” GAO/AIMD-00-209R, Jun. 15, 2000.

U.S. General Accounting Office, “Electronic Government: Federal Initiatives are Evolving Rapidly But They Face Significant Challenges,” GAO/T-AIMD/GGD-00-179, May 22, 2000.

U.S. General Accounting Office, “Electronic Government: Government Paperwork Elimination Act Presents Challenges For Agencies,” GAO/AIMD-00-282, Sep.15, 2000.

U.S. General Accounting Office, “Federal Chief Information Officer: Leadership Needed to Confront Serious Challenges and Emerging Issues,” GAO/AIMD-00-316, Sep.12, 2000.

U.S. General Accounting Office, “High-Risk Series: An Update,” GAO/HR-99-1, Jan. 1999.

U.S. General Accounting Office, “High-Risk Series: Information Management and Technology,” GAO/HR-97-9, Feb. 1997.

U.S. General Accounting Office, “Information Security: Serious and Widespread Weaknesses Persist at Federal Agencies,” GAO/AIMD-00-295, Sep. 06, 2000.

U.S. General Accounting Office, “Information Security: Software Change Controls at the Department of Defense,” GAO/AIMD-00-188R, Jun. 30, 2000.

U.S. General Accounting Office, “Major Management Challenges and Program Risks: Department of Defense,” GAO 01-244, 2001.

U.S. General Accounting Office, “Major Management Challenges and Program Risks: Department of Defense,” GAO/OCG-99-4, Jan. 1999.

Verlage, M., “Towards Software Process Models,” in Proceedings of the Tenth International Software Process Workshop, IEEE (Dijon, France, 17-19 June, 1996), pp. 112-114.

Vigder, M., and Kark, A., “Software Cost Estimation Study,” National Research Council Canada, Ottawa, Canada, 1994. [http://wwwsel.itt.nrc.ca/Projects/cp/ NRC37116/ Chap1.html]. Accessed Mar. 2003.

Voas, J., and Agresti, W., “Software Quality From a Behavioral Perspective,” *IT Pro*, July-August 2004, pp. 46-50.

Wallace, D., Peng, W., and Ippolito, L., "NISTIR 4909 Software Quality Assurance: Documentation and Reviews," U.S. Department of Commerce, [http://hissa.ncsl.nist.gov/publications/nistir4909/], Jan 2001. Accessed Jul. 2004.

Wallin, C., Ekdahl, F., and Larsson, S., "Integrating Business and Software Development Models, *IEEE Software*, November/December 2002, pp. 28-33.

Wang, W., and Kececioglu, D., "Confidence Limits on the Inherent Availability of Equipment," in *Proceedings of Reliability and Maintainability Symposium*, IEEE (Washington DC., Jan. 2000), pp. 162-168.

Ward, W., and Venkataraman, B., "Some Observations on Software Quality," in *Proceedings of the 37th Annual Conference on Southeast Regional Conference*, ACM (Mobile AL, Apr. 1999) pp. 1-9.

Wheeler, S., and Duggins, S., "Improving Software Quality," in *Proceedings of the 36th Annual Conference on Southeast Regional Conference*, ACM (Marieta, GA., Apr. 1998) pp. 300-309.

Wiegers, K., *Software Requirements Second Edition*, Microsoft Press, Redmond, 2002.

Wohl, A., "Pricing for Success in the ASP Market", [http://www.wohl.com/wa0130.htm], Jan 2001. Accessed Jul. 2004.

Wong, S., "Software Acquisition Management Experience Learnt in a Multi-Discipline and Multi-Contract Project Environment," in *Proceedings of the first Asia Pacific Conference on Software Quality*, IEEE (Hong Kong, Oct. 2000) pp. 239-247.

Wysocki, R., Beck, R., and Crane, D., *Effective Project Management*, 2nd ed., Wiley Computer Publishing, New York, 2000.

Xia, F., "What's Wrong with Software Engineering Research Methodology," *ACM SIGSOFT Software Engineering Notes*, Vol. 23, No. 1, pp. 62-65, Jan. 1998.

Yourdon, E., *Decline and Fall of the American Programmer*, Prentice Hall, Upper Saddle River, 1993.

Zahniser, R., "Building Software in Groups," American Programmer, Vol. 3, Nos. 7-8, Jul-Aug 1990.

Zultner, R., "Quality Function Deployment for Software: Satisfying Customers," American Programmer, February 1992, pp. 28-41.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. VADM Keith Lippert
   Defense Logistics Agency
   Fort Belvoir, Virginia

4. Mr. Douglas Verhagen
   Naval Supply Systems Command, CIO Staff
   Mechanicsburg, Pennsylvania

5. CAPT Juan Nogueira
   Uruguay Navy
   Montevideo, Uruguay

6. Ms. Charito Glorioso
   Defense Contracting Command
   Washington, D.C.

7. CAPT John H. Chase Jr.
   Chief of Naval Operations (N781)
   Washington, D.C.

8. Ms. Mary T. O'Hara
   U.S. Army PEO Enterprise Information Systems
   Fort Belvoir, Virginia

9. CDR Dave Hellman
   Chief of Operations, Navy
   Washington, D.C.

10. Mr. Andrew Christensen
    NAVSUP East Coast NMCI CTR
    Norfolk, Virginia

11. Dr. Beryl Harman
    Defense Acquisition University
    Fort Belvoir, Virginia

12. Dr. Bret Michael
Naval Postgraduate School
Monterey, California

13. Dr. Dan Boger
Naval Postgraduate School
Monterey, California

14. Dr. Man-tak Shing
Naval Postgraduate School
Monterey, California

15. Dr. John Osmundson
Naval Postgraduate School
Monterey, California

16. Professor Rex Buddenberg
Naval Postgraduate School
Monterey, California