A Software Assessment of a Small Organization

by

Binh-Minh Tran

Project Report submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

Approved:

Sallie M. Henry

Committee Chairperson: Sallie M. Henry

[illegible]

Dennis G. Kafura

[illegible]

James D. Arthur

December, 1995

Blacksburg, Virginia

c.2

LD
5655
V851
1975
T736
c.2

A Software Assessment of a Small Organization

by

Binh-Minh Tran

Committee Chairperson: Sallie M. Henry
Computer Science

Abstract

This report describes a software development process assessment performed at a small organization which is part of the Department of Defense. The assessment follows the guidelines of the Software Engineering Institute's Capability Maturity Model. The procedure used for the assessment is described in detail in this report. Results obtained from each step in the assessment are illustrated and analyzed. The report concludes with a set of recommendations for improving the organization's software development process.

Acknowledgments

I would like to thank many people for making it possible for me to complete this work.

Dr. Sallie Henry for providing the valuable advice that I needed to complete the work and for encouraging me when I was very discouraged.

My parents for always believing in me and often encouraging me by telling me how they are proud of me.

My sisters, Lieu Nhu and Diem Tuyet, for calling me up on the phone and reminding me that I am almost there and not to be discouraged.

All my friends, for proofreading and offering suggestions to improve this manuscript.

The studied organization, for allowing me to perform the assessment at its facility.

Everyone at my place of employment, particularly, Bob Crowder, Ken Novell, Jacqueline Hart, Sandi Cullen, and Greg Murphy for giving me the opportunity to complete my degree.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1. Introduction

Introduction

For many years, software development has been described as a process that is a mixture of science and art. Statements like this are the result of the fact that the outcome of a software project is greatly influenced by the people who perform the development and the process they use. Given a set of requirements, ten different programmers can produce ten different solutions. The reason for this is that programming is affected by the authors' personality, ability, and experience, and because software development does not have a well-defined discipline as do other engineering fields. Software projects are developed in such a human-personality-dependent way are prone to errors. Erroneous software is costly, particularly software that is used in critical systems that involve human life or the national security.

Today, with the increase in user expectation and the rapid advance of computer hardware technology, the requirement for sophisticated software is increasing, yet there has not been as rapid an increase in our ability to develop these software systems. Jensen and Tonies illustrate this situation by the graph shown in Figure 1-1. In order to reduce the imbalance, a better method to build software must be produced.

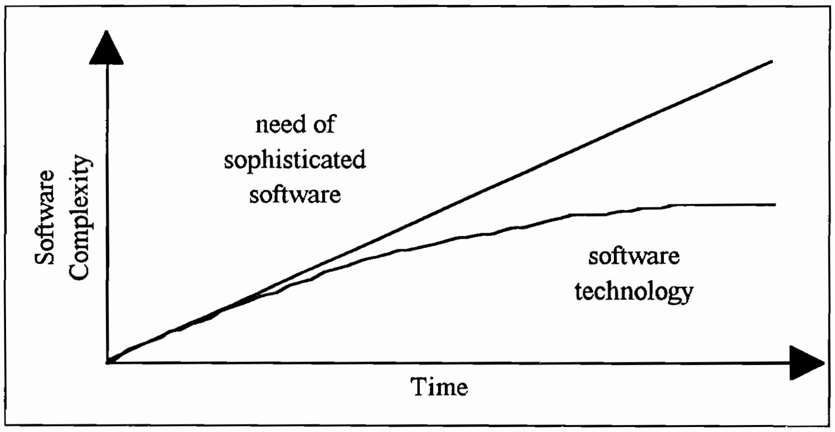


Figure 1-1. Imbalance of requirement and technology of sophisticated software.

The increase in complexity of software makes correctness and reliability become more difficult to accomplish, yet these issues remain vitally important. Studies have shown that software developed without careful design tends to contain errors and omissions that are not detected until close to delivery time. At delivery time, programmers, often with their managers' approval, tend to patch code and eliminate requirements that might be crucial to the customers. Patched software is difficult to maintain. Brooks has shown that the cost of maintenance is generally about 60% of the entire development cost [BROF82]. Eliminating customer requirements reduces customer satisfaction and may cause the customers to reject the software completely.

So there are two apparent problems:

- We need to study the software development process so that we can make improvements that will enable us to satisfy the need for more complex software.
- We need to be sure that, until better methods are developed, we are at least using state-of-the-art software development technologies.

The first step in solving each of these problems is to develop a method to assess the software development process, and to apply that method to a number of organizations.

This report presents a study in which the software development process of an organization is assessed and a set of recommendations is made to improve its process. The next section explains the necessity of an assessment in the process of improving an organization's software development process. The section following the rationale describes the model that is used as a guideline for the assessment in this study. The section after that explains the motivation of this work. The last section summarizes the chapter and provides the layouts of the subsequent chapters in this report.

Necessity of an Assessment

In recent years, many efforts have been made to develop a discipline in which the software development process can be managed in an orderly way. The goals have been to reduce cost, to meet the schedule, and to deliver products that meet all the customer requirements. Unfortunately, "one size does not fit all" [BOND95]. According to David Bond, each organization needs a set of practices that work best for that organization. Furthermore, since the nature of the projects at an organization varies, once a process has been defined for software development at an organization, it must be modified with careful considerations to fit each project.

In order to establish a set of practices that are appropriate for its software development process, an organization needs to understand the current status of its process. The Software Engineering Institute (SEI) has developed a software-process capability maturity model, which can be used by any software organization to assess its software development process and identify the areas most in need of improvement [HUMW88]. The assessment determines the maturity level of the software development process employed at the organization. This allows the organization to identify areas in need of improvement, and to identify methods for improvement. The basic objectives of an assessment are to:

- learn how the organization works,
- identify its major problems, and
- enroll its leaders in the change process[HUMW91].

This assessment has two parts: the initial survey and the follow-up interview. Because some projects involve classified material, this study can not include the review of those projects. Therefore, the follow-up interview is more crucial for a thorough understanding of the software development process.

SEI Capability Maturity Model

The Capability Maturity Model (CMM) is a framework that was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University. This framework attempts to quantify a software organization's capability to "consistently and predictably produce high-quality software products" [SAIH95]. Its purpose is to assist an organization in establishing the goals to improve the software development process and to track the improvement of that organization. SEI defined the five levels of maturity and developed a questionnaire which is used to determine the capability level of an organization.

The questionnaire is used to collect information about how the organization currently performs its software development. From the analysis of the collected data, one of the five SEI maturity levels can be assigned. Knowing where the organization is on the maturity scale assists software professionals and managers in recognizing the areas that are most in need of improvement.

The five levels of CMM are identified as: Initial, Repeatable, Defined, Managed, and Optimizing. Humphrey et al. discuss the reasons SEI chose these maturity levels [HUMW89b]. SEI believes that those levels:

- reasonably represent historical phases of evolutionary improvement,
- provide achievable improvement steps in a reasonable sequence,
- suggest interim improvement goals and progress measures, and
- provide immediate improvement priorities once an organization's status in this framework is known.

In this model, the lower levels are characterized by the capabilities that are the foundations for an organization to progress to the higher levels. The characteristics of the five maturity levels are described in the following subsections.

1. Initial

The software development process of an organization at this level is an "ad hoc and chaotic process" [HUMW91]. Typically, a project is carried out with no formal procedure, no cost and schedule estimates, and no project plan. This makes it difficult for management to control the process. The lack of tools or the lack of use of tools also plays a major role in preventing the management from controlling the development process and ensuring that procedures are followed. The most noticeable characteristic of an organization at this level is the senior management's lack of understanding of the key issues in software development, which results in a lack of motivation for process improvement and procedure enforcement.

2. Repeatable

The process at this level "provides control over the way the organization establishes its plans and commitments" [HUMW89a]. Thus, a level-2 organization has established the basic project control so that cost and schedule estimates can be made with some accuracy on a consistent basis. However, according to Humphrey, the process at this level is capable of repeating similar works but struggles when facing new challenges. He also advises that introduction of new tools, particularly those for automatically managing the process, should be carefully considered because although these tools can be effective, introducing them at this level may cause chaos. Furthermore, an orderly framework for process improvement is not defined at this level, thus, a major change in the organization structure can cause disruption in the improvement process.

3. Defined

At this level, an organization has established groundwork on which its professionals can continue to improve their process. A Software Engineering Process Group (SEPG) is formed and helps the organization realize its process improvements. The two main tasks of this group are initiating and sustaining process change and supporting normal operations. When incorporating new methods or updating current software practices, the process group must ensure the effectiveness of these changes. When supporting the normal operations of the organization, the responsibilities of the process group is to define the development process that the organization can follow, to identify technology needs and opportunities such as automated tools and technical training, to guide the projects in the right directions, and to conduct quarterly management reviews of process status and performance. To avoid increasing the contention often introduced by the traditional conflict between the Software Quality Assurance and the software development, the SEPG must not report to either leader of these two groups. Humphrey recommends that the SEPG report to a higher level of management.

4. Managed

A level-4 organization can actually measure its process to ensure continued control in crisis situations. Quantitative quality goals are set for both software products and processes so that a quality product is provided within the time limits. In situations where deadlines are missed, some actions must be taken to study and understand the mistakes and prevent their reoccurrence. Data gathering and maintaining at this level is controlled with an organization-wide software process database. The data are treated as a resource that is used to analyze process measurements and to improve the software process.

5. Optimizing

A level-5 organization has optimized its software development process. The optimization is a continuous process throughout the organization. The automation of data collection is realized. The data are used to identify the weakest process elements and activities where optimization is possible. The main goal at this level is to prevent the reoccurrence of defects by carefully and thoroughly analyzing the defects to recognize their causes. Any means found to improve the process are incorporated into the ever-evolving process.

SEI has defined the internal structure of the CMM hierarchically as illustrated in Figure 1-2 (SEI95). The structure depicts the decomposition of each maturity level into several Key Process Areas (KPAs), each of which contains a set of key practices. The key process of a level identifies areas which the software development process of an organization must satisfy to achieve that level. Since level 1 is the lowest level, there is no requirement for achievement at this level; thus, it does not have any KPAs. A summary of the KPAs for each maturity level is provided in Figure 1-3 [SEI95]. Each KPA is described in terms of key practices which identify the essential elements for making the software development process effective. These key practices are intended to describe what needs to be done, but not how the goals are to be accomplished. The implementation of a key practice depends on each project and should be tailored to be most effective. The summary of these key practices, obtained from [SEI95], are presented in Appendix C.

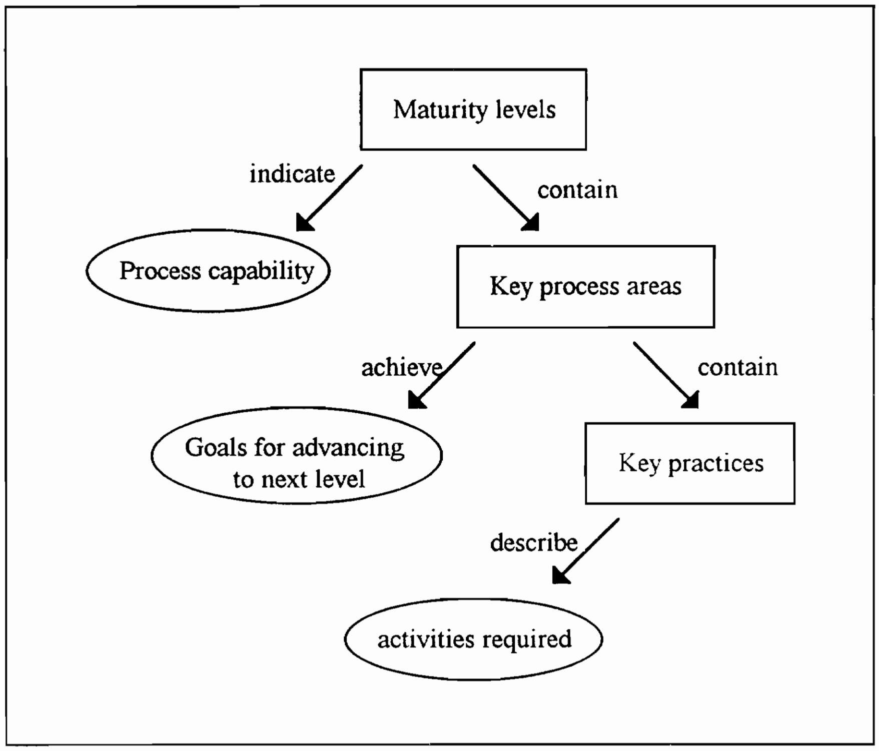


Figure 1-2. The CMM Structure [SEI95].

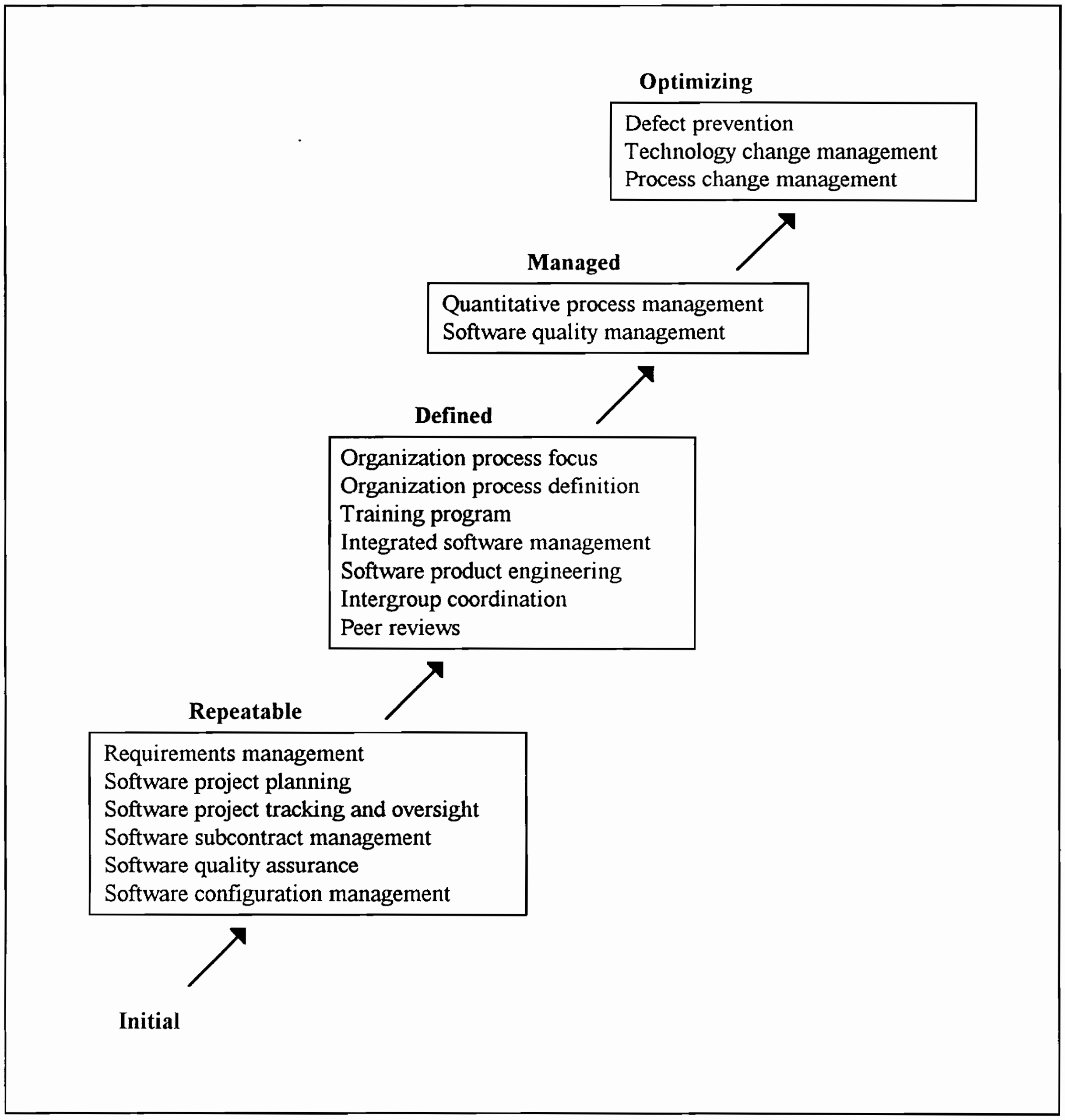

Optimizing
Defect prevention
Technology change management
Process change management
Managed
Quantitative process management
Software quality management
Defined
Organization process focus
Organization process definition
Training program
Integrated software management
Software product engineering
Intergroup coordination
Peer reviews
Repeatable
Requirements management
Software project planning
Software project tracking and oversight
Software subcontract management
Software quality assurance
Software configuration management
Initial

Figure 1-3. The key process areas by maturity level [SEI95].

Motivation of this Study

The need for quality software has been dramatically increasing. Many efforts have been made to change the process of software development to a more systematic methodology in which costs are kept lower, schedules are met, and customers are more satisfied. These efforts have significantly influenced both industry and academics. The Software Assessment course offered at Virginia Polytechnic Institute and State in the Spring of 1995 introduced students to the SEI software development assessment process. In completing the course, all students were involved in the process of assessing the software development process at one of two small local companies. Students were divided into teams of 2 to 4 people. Each team performed a part of the assessment. One of the organizations studied was Recognition Research Incorporation (RRI) in Blacksburg. At the end of the course, the students presented the results of the assessment of the current development process at RRI and provided a set of recommendations. Interest in the results at RRI was the motivation for the study described in this report.

This study provided me with the experience of conducting a complete assessment myself as well as an in-depth view of how two very different organizations produce software.

Summary

A software development organization must continue to improve its software development process to remain competitive. In order to make an appropriate plan for improvement, it is necessary for the organization to understand its own status and capabilities. Thus an assessment is a first and important step for the organization to make in its progress toward improvement. In this report, Chapter 2 summarizes several related works being performed in the area of software development assessment and improvement. Chapter 3 briefly describes the background of the studied organization. Chapter 4 lays out the specific steps that are performed in the assessment. Chapter 5 and Chapter 6 present the two primary parts of the assessment: the Initial Assessment and the Follow-up Assessment. Chapter 7 provides the highest-priority recommendations that are produced based on the results of the assessment. The last chapter outlines the conclusions from the study.

Chapter 2. Related Work

Introduction

Many organizations have employed the SEI Capability Maturity Model to set goals for improving their software development process. Over the last decade, studies have shown that many of those organizations have successfully achieved higher SEI levels and continued to improve [DIOR92][HUMW91][JOHA94]. Although some organizations are not willing to reveal their assessments, others have contributed valuable information to the software engineering literature by reporting the results of their experiments. This chapter briefly describes how several software organizations have used the SEI framework to accomplish their software development assessments and improvements. These assessments and improvements were performed at Hughes Aircraft's Software Engineering Division, Raytheon's Equipment Division, and at Corning Incorporated's Information Services Division.

Software Assessment at Hughes Aircraft's SED

In 1987, a software development assessment was conducted at Hughes Aircraft's Software Engineering Division (SED) by SEI [SAIH95]. At this assessment, SED was found to be at level 2 of the SEI framework's scale. The assessment team provided SED with seven recommendations for improvement in the following areas: quantitative process management, process focus, requirements management, software quality assurance, training program, technical practices, and working relationships. SED established a software improvement plan which laid out five improvement tasks as the process-requirement specifications. These five tasks are listed below:

- Form a software-engineering process group
- Implement quantitative process management
- Fill in the gaps in training
- Standardize an effective review process
- Move toward a software-engineering discipline

Two recommendations in the areas of quality assurance and working relationships were not included in the plan since they were outside the scope of SED. The organization put the plan to work and "found that the investment improved working conditions, employee morale, and the performance as measured in project schedule and cost" [HUMW91].

Another assessment was performed at SED in 1990 at the request of Hughes. The organization was found to be a strong level 3. Furthermore, it had also carried out activities that were required to be in levels 4 and 5. As shown, the processes of software development assessment and improvement at Hughes were very successful.

Software Assessment at Raytheon's Equipment Division

Raytheon had a software assessment performed within its Software System Lab in early 1988 [DIOR92][SAIH95]. At this assessment, they were found to be at slightly below level 2, and they identified four areas that needed improvements: documented practices and procedures, training, tools and methods, and metrics.

They immediately began to establish the plan for their software development process improvements and realized it. These improvements have greatly benefited the company. The organization tracks its improvements based on Phil Crosby's Cost of Quality idea [CROP84] in which the cost of rework is measured against the cost of doing it right the first time. The measure has shown that the rework costs have substantially decreased, resulting in the savings of $9.2 million on the nearly $115 million of software-development costs. In 1992, after a follow-up analysis, the organization was assessed at level 3 and was on its way to level 4 [DIOR92]. The experience at Raytheon's Equipment Division provided another example of success in employing the SEI model to improve an organization's software development process.

Software Process Improvement at Corning Incorporated

Corning Incorporated used Software Process Assessment as part of its Information Services Division's (ISD) software process improvement program [JOHA94]. At the beginning of 1991, the organization followed the guidelines provided by the SEI to perform an assessment in the ISD's software development process. This assessment was to identify and assign priorities to key areas for process improvement, and to propose a framework for subsequent improvement. The assessment findings were combined with the organization's current practices to establish the recommendations which called for improvements in project planning and execution and in key aspects of the software development cycle. The definition of their improvement plan was then established. It involves:

- Defining their "current best practice" model;
- Benchmarking and technology scan;
- Defining their desired software process[JOHA94].

This improvement approach is illustrated by [JOHA94] as shown in Figure 2-1 below. The plans are carried out in the order that they are specified. The output of the previous step is used in the following step.

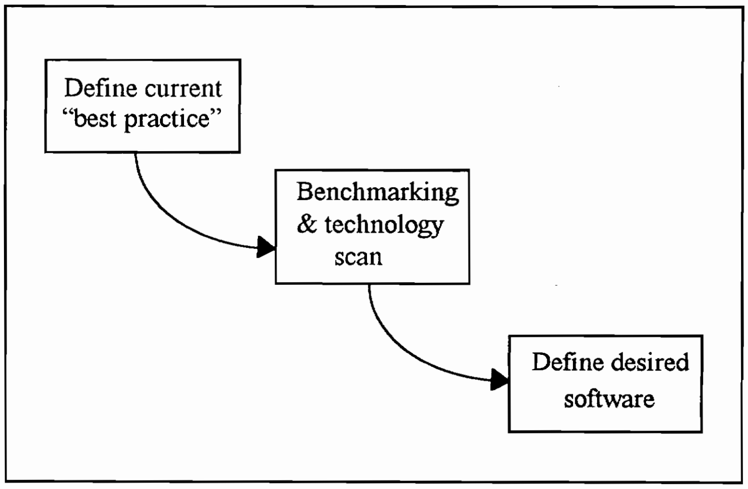

Define current "best practice"
Benchmarking & technology scan
Define desired software

Figure 2-1. Improvement approach at ISD.

Summary

Although software process improvement does not return immediate revenue, many organizations are willing to invest their money, time, and effort in attempting to improve their process. These organizations must understand and accept the delay in return of benefits from the improvement. They try to learn about their software process, obtain and realize recommendations for improvement, and gradually progress up the capability maturity scale. Many organizations have greatly benefited from the assessment process. Many others have just started their improvement process. The rest of this document describes the attempt of a small organization to improve its software development process.

Chapter 3. Organization's Background

Introduction

The organization in this study prefers that its name not be revealed. Thus, for convenience, the organization is referred to as XYZ in this document. XYZ is a small organization within the Department of Defense (DOD). XYZ is interested in performing an assessment to help its management and engineers improve their software development process. The management wants to emphasize the identification of methods for improvements. Since the organization prefers not to release its identity, the information on its background must be limited. Furthermore, concerns about security also limit what can be provided in this chapter. However, some general information about XYZ such as the organization's structure and the non-specific domain of its projects are presented in the next sections.

XYZ's Organizational Structure

XYZ is a relatively small group of professionals that includes both in-house government employees and contractors. As with many other organizations in the DOD, their work is sponsored by the Naval Sea System Command (NAVSEA) in Washington, D. C. The organization also receives commands and approvals from NAVSEA. The relationship between the organization and its sponsor requires periodic reviews of progress, quality, schedule, and budget of the sponsored projects.

As mentioned above, XYZ is a small organization. The total number of employees involved in this study is 17, 7 of whom are government employees. Figure 3-1 shows the organizational structure of XYZ. The development manager is the leader of five groups that include both in-house employees and contractors. Three of these groups involve software development. Each group is responsible for one or more projects. As shown in the figure, the in-house employees and contractors report to more than one project leader, which means one person might be assigned to more than one project. Also under the development manager is a group of contractors who are responsible for independent verification and validation (IV&V). The last group under the development project is the configuration management/quality assurance (CMQA). The team members often work closely in a limited access computer area to perform their software development. Although there is a difference in employment status, the in-house employees and the contractors are heavily involved in all the software development activities at XYZ. Therefore, the contractors have been included in this study. The next sections provide a general description of the work performed by XYZ.

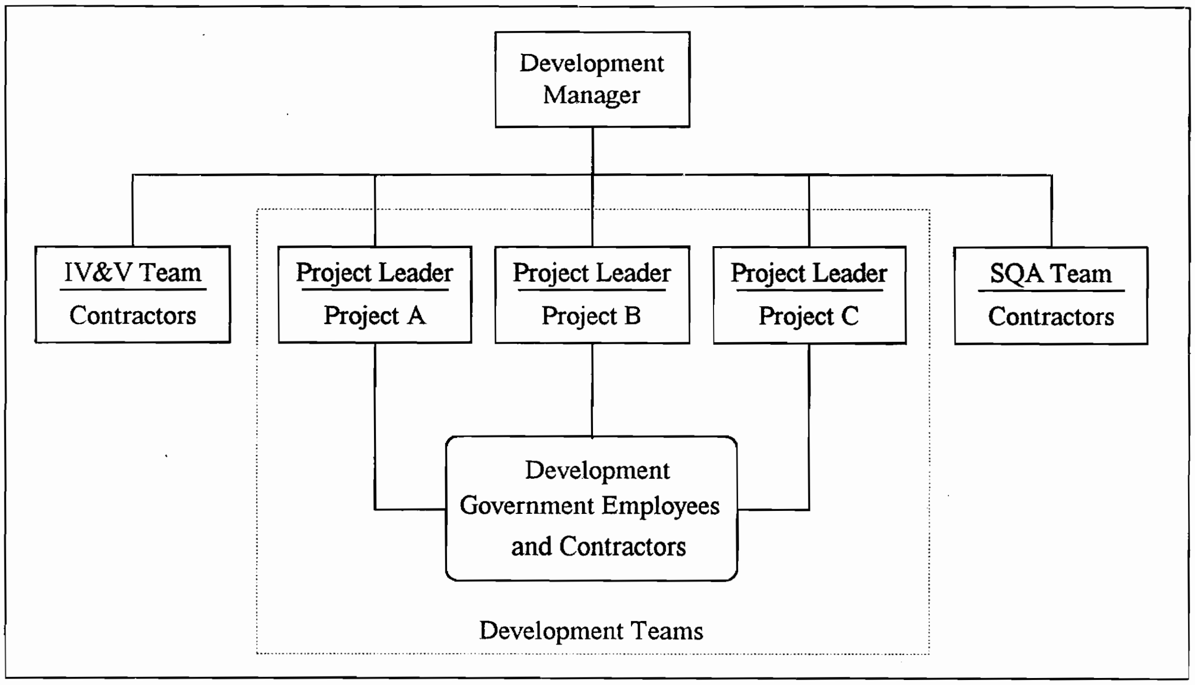


Figure 3-1. The organizational structure of XYZ.

XYZ's Project Nature

XYZ's projects involve the development of some of the computer programs that support the U.S. warships under NAVSEA command. To ensure that no classified information is revealed, specific descriptions about XYZ's projects are not provided.

The main responsibilities of the organization include developing programs, installing the programs on the ships, and providing ongoing technical support. Rather than developing programs strictly from requirement specifications, XYZ often either maintains programs that are prototypes produced by another organization or updates programs that are of previous baselines. A prototype is designed based on customer requirements which are specified at a very high level. The prototype is tested on the ship by the organization that produced the prototype. However, the testing is not thorough, since its main purpose is to demonstrate basic capabilities to the customer. XYZ engineers next generate requirements that are based on the original customer requirements but are more specific and complete. XYZ then modifies the prototype to satisfy the new requirements and delivers the modified prototype to the ships. After the system is installed on the ships, it is maintained by XYZ.

XYZ also provides upgrades to different programs. This kind of work is performed more frequently than development because the organization builds many different baselines from a single prototype, depending on the types of ships that are supported. Very infrequently, XYZ develops new programs without being provided with a prototype. Their main work is more maintaining than developing.

Summary

XYZ is attempting to realize the improvement with the assistance of the SEI software assessment model. Due to the small size of the organization, the personnel resources they can commit to the improvement process are very limited. With the understanding and support of the management, XYZ is trying to put effort into improving its software development process. Therefore, the goal of this study is to learn about the organization's software development process and provide some recommendations so that its management can start making plans for improvement.

Chapter 4. Assessment Procedure

Introduction

This chapter describes how the assessment process was conducted at XYZ. The assessment process consisted of two stages, the Initial stage and the Follow-up stage. The activities in the initial stage included collecting the survey data, analyzing the survey data, generating follow-up questions, and selecting the employees for the follow-up interview. The activities in the follow-up stage included conducting the follow-up interview, analyzing the follow-up interview results, and developing the set of recommendations for improvement. An overall procedure is discussed in the next section. The section followed that describes in detail the activities of the two assessment stages.

Assessment Procedure at XYZ

The CMM and the questionnaire were developed by SEI with the intention to help a company who wishes to improve its software development process. However, this model is more suited for large organizations than for small ones. The questionnaire contains many questions that focus on aspects which are more applicable to large companies and often not to small companies. Followings are the examples of several questions which involve aspects which small organizations often are not capable of doing often because of the lack of personnel and budgets:

- Question #23:

  For each project, are independent audit conducted for each step of the software development process?

  Independent audit is usually expensive, thus, small organizations often do not look into this activity.

- Question #34:

  Are software staffing profiles maintained of actual staffing versus planned staffing?

  Since with small organizations, personnel resource is low, the management often reappoint staffs without maintaining profiles about the planned and actual appointing.

- Questions:

  #40: Are target computer memory utilization estimates and actuals tracked?

  #41: Is target computer I/O channel utilization tracked?

  Small organization with low number of engineers do not often have problem with memory and I/O channel utilization and even though it does occur, the engineers often can quickly resolve. Thus, it is not necessary for the management to estimate and record the actuals of target computer memory and I/O channel utilization.

- Question #91:

  Are the software development and maintenance personnel provided with interactive documentation facilities?

  Small organizations usually can not afford this type of facilities.

- Questions:

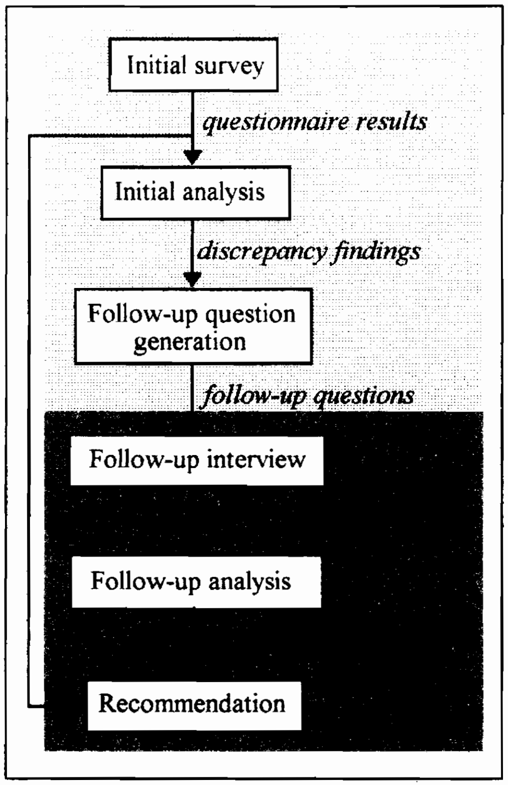
  #93: Are prototyping methods used in designing the critical performance elements of the software?

  #94: Are prototyping methods used in designing the critical elements of the man-machine interface?

  Prototyping is often too expensive for a small organization, especially, when the organization has a good practice of throwing away the prototype.

With the existence of the above areas, it is more difficult for the questionnaire to truly reflect the actual status of a small organization by simply identifying a maturity level based on the answers from the questionnaire. Thus for a large organization, as intended by its developer, the SEI questionnaire can be used to identify a maturity capability level of the organization's software development process. However, for a small organization, the questionnaire should be used to identify areas where further study about its software development need to focus on. The further study is accomplished with the follow-up interviews which are performed with several selected employees.

The assessment process then includes the main activities shown in Figure 4-1. These activities are carried out in the order presented. Those that belong to the initial assessment stage appear at the yellow area and the blue area presents the follow-up assessment stage. Each succeeding activity takes the data generated from the previous activity and produced data for the next one. As shown in the figure, the data from the first activity, the initial survey, and the results from the follow-up analysis are both inputted into the development of the recommendation set. This illustrates that the questionnaire itself is not sufficient for the assessor to fully understand the development process of a small organization and thus, the follow-up interview is necessary and, as later discussed throughout this document, provide a



substantial amount of information about the software development process of the organization in this study.

Each of these activities is discussed in details in the following section.

Figure 4-1. Assessment procedure at XYZ.

Initial Assessment

The initial assessment is accomplished by the following four activities.

1. **Collecting initial survey data**

   The questionnaire is provided to each employee. A copy of this questionnaire is provided in Appendix A. While answering the questionnaire, the employees are not allowed to communicate with each other. The questionnaire takes approximately 25 minutes for each employee to complete. The results are collected and used as input to the analysis step below.

2. **Analyzing the initial survey data**

   The collected data are examined using the spreadsheet application Microsoft Excel. The percentage of respondents answering "yes" to each question is computed. Questions that cause respondents to be evenly divided are noted. These questions indicate that the employees do not fully understand the organization's software development practices or that the employees have misinterpreted the terminology used in the questionnaire. Areas that are related to these answers are identified and described in Chapter 5, the Initial Survey Analysis. A statistical analysis and some significant findings from the initial survey data are provided in the chapter. This analysis provides some insights into different patterns found in the initial survey data.

3. Generating the follow-up questions

Because the answers to the questionnaire may indicate some confusion and some disagreement among the employees regarding the organization's software development activities, a follow-up interview is usually necessary. The follow-up interview is intended to clarify any misconceptions or ambiguities by providing both the interviewees and the interviewers with the opportunities to explain uncommon situations and to clarify cases where answers were affected by misinterpretation of the questionnaire terminology. The analysis of the initial survey data is used to develop the follow-up questions. These questions focus on the identified areas of conflict described in the previous section. The follow-up questions are then used to interview the employees who are selected as described in the next subsection.

4. Selecting the interviewees

The selection of the interviewees is not based on any defined number or rules but is based on the employees' answer patterns and their positions in the organization. One of the management level personnel is chosen so that some insights from the management can be obtained. Two other people whose answer patterns are very positive and very negative are selected to be interviewed because their points may confirm or clarify the discrepancies occurring in the initial findings. Another person who has many answers that are opposite from the majority is also chosen. Two people whose answers are very mixed are selected. A total number of six employees are chosen for follow-up interview. One of the selected employees holds a management position. The rest include one government employee and four contractors.

Follow-up Assessment

The follow-up assessment includes the following three activities.

1. **Interviewing the selected employees**

   Each follow-up interview generally lasts between 25 and 40 minutes. During the interviews, the follow-up questions are used to stimulate the dialog. Then, based on the interviewee's answers, the interviewer generates new questions to obtain a better understanding of a specific area. An important part of the interview occurs when the employees ask the interviewer to clarify some terms used in the questions. The employees also explain several uncommon situations which would not have been possible to explain without the interview. The answers in the interviews are noted and analyzed in the next step.

2. **Analyzing the follow-up interview results**

   After the interviews, answers from each employee are analyzed. The analysis of the follow-up interviews is provided in Chapter 6. An interesting point observed from these interviews is, in general, the answers of all the interviewees seem to be consistent with each other. The fact that, although the interviewees are selected based on their inconsistent initial answers, the results from the follow-up interviews provided a consistent picture has implied that the follow-up interview is an effective procedure in the assessment process. It successfully clarified many misinterpretations and misconceptions that affected the results of the initial survey.

3. **Developing the recommendations**

   When the analysis of the follow-up interview is complete, a set of recommendations is generated. The development of the recommendations is based on findings from the initial analysis, the improved understanding from the follow-up interview, and the referenced materials on software engineering.

Summary

The activities described in this chapter are performed during the assessment process of the studied organization. A summary of the assessment process is provided in Figure 4-1. This figure lists the steps in the sequential order of which the steps are carried out. All of the data obtained from the initial survey and the follow-up interview are analyzed carefully. This analysis is presented in detail in Chapter 5 and Chapter 6.

Chapter 5. Initial Survey Analysis

Introduction

The initial survey is conducted and the results are collected. The data is entered into electronic form using the Microsoft Excel spreadsheet. The percentage of respondents answering "yes" to each question is computed and recorded. Questions resulting in approximately 50% "yes" answers are considered to be indicators of discrepancies. This chapter covers the analysis of the initial survey data. The analysis consists of the presentation of the significant findings and any insights drawn from the findings.

Initial Analysis

As the collected results from the questionnaire are reviewed, several significant results are found. On some questions, 100% of respondents answer "yes". This yields high confidence that the activities corresponding to these questions are being performed at the organization. Also of significance are the situations where about one half of the employees respond "yes". These discrepancies may result from either misinterpretation of the questions or uncertainty about how the organization functions in particular areas. Another significant result is the highly disparate responses, such as very positive or very negative from one person. For the two latter cases, a follow-up interview is necessary to provide a thorough understanding of the current software development activities at XYZ.

The questions to which more than 80% of the employees answer "yes" are considered to correspond to activities performed at XYZ. Table 5-1 on the next page provides the questionnaire results. The first column shows the percentage of the employees answering "yes" to a question or a group of related questions. The second column shows the areas or activities emphasized in the questions. Figure 5-1 shows an interesting comparison between the answers of the leaders and the answers of the engineers. To these questions, all supervisors answer "yes" but some engineers do not agree. The difference is, however, relatively small. This small disagreement might be a result of the difference in responsibilities and perception of the activities between the leaders and the engineers, because the questions shown in this figure mostly involve management areas. Thus, XYZ has performed some of the activities of interest to SEI. The number of these activities, however, is still small -- 10% of all activities. Out of this small number, most of the activities performed are at level 2 and none are higher than level 3. This indicates that XYZ has an unpredictable and uncontrolled software development process.

The analysis next focuses on the questions in which disagreement among employees occurs. Table 5-2 provides the areas and there corresponding questions to which only about half of the employees answer "yes". These areas are the focus of the follow-up questions.

Table 5-1. Questions to which more than 80% employees answer "yes".

% "yes"	Areas or activities	Questions
100	There is a designated software manager for each software development project and this manager reports directly to the program manager.	1, 2
82	Software system engineering is represented on the system design team.	5
94	There is a software configuration control function for each software development project.	6
82	A formal procedure is used to assure periodic management review of the status of each software development project.	20
90	Action items and trouble reports resulting from design reviews, code reviews and testing are tracked to closure.	44, 45, 46
88	There are mechanisms for controlling changes to the software requirements.	66
94	Internal software design reviews are conducted.	69

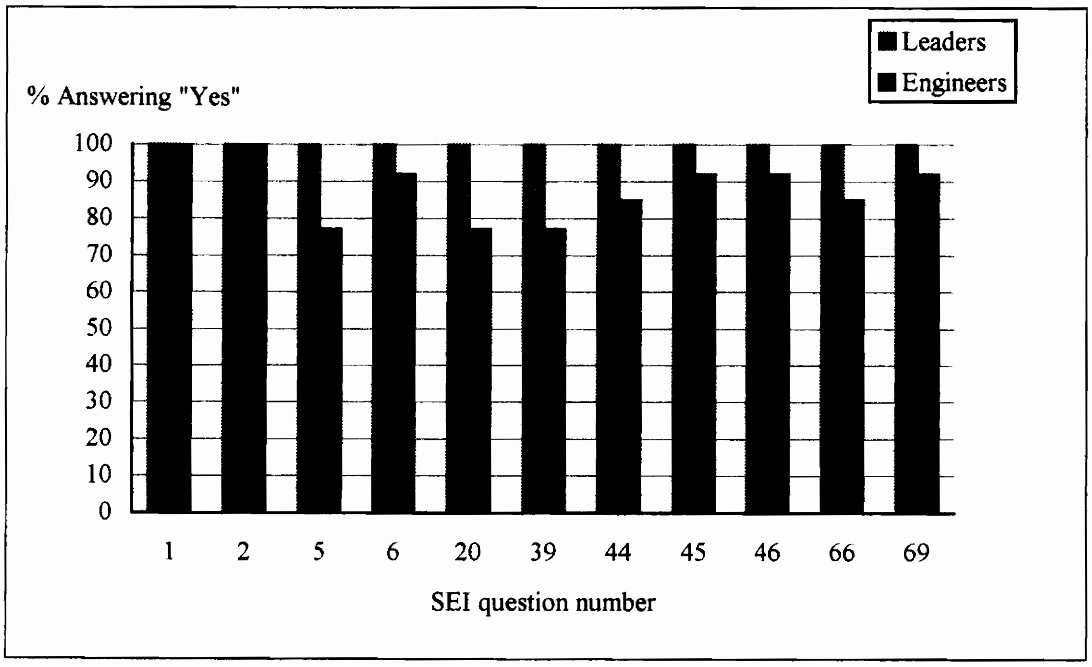


Figure 5-1. All leaders answer "yes" to the 80%-answering-"yes" questions.

Table 5-2. Areas in which discrepancies occur and are clarified.

Areas or activities	Questions
1. applying standards to the development activities	18, 19, 22, 26, 28, 29, 63
2. conducting formal reviews for each stage of software development	73, 79
3. use of data gathering during reviews in software development	42, 43, 50, 52, 53, 72,
4. mechanisms used in technical interchanges	60, 61, 62
5. mechanisms for configuration management and code changes	75, 76, 80
6. use of automated tools for software development	65, 68, 81, 83, 85, 89, 92

In addition, the number of questions to which all the leaders agree is far more than the number of questions to which all the engineers agree, both "yes" and "no" -- 32 and 4, respectively. This might be resulted from the lack of distributing information among the engineers and from the management level personnel to the engineers. Of the 32 questions in which the leaders agree, 22 are answered "yes". This imbalance is shown in Figure 5-2 and Figure 5-3. Figure 5-2 presents the questions associated with CMM level 2 and Figure 5-3 presents those that are associated with levels 3, 4, and 5. The high number of "yes" answers from the management might be explained by the fact that many of the questions included in these figures involve with management level work. Thus it is often not well-known to many engineers.

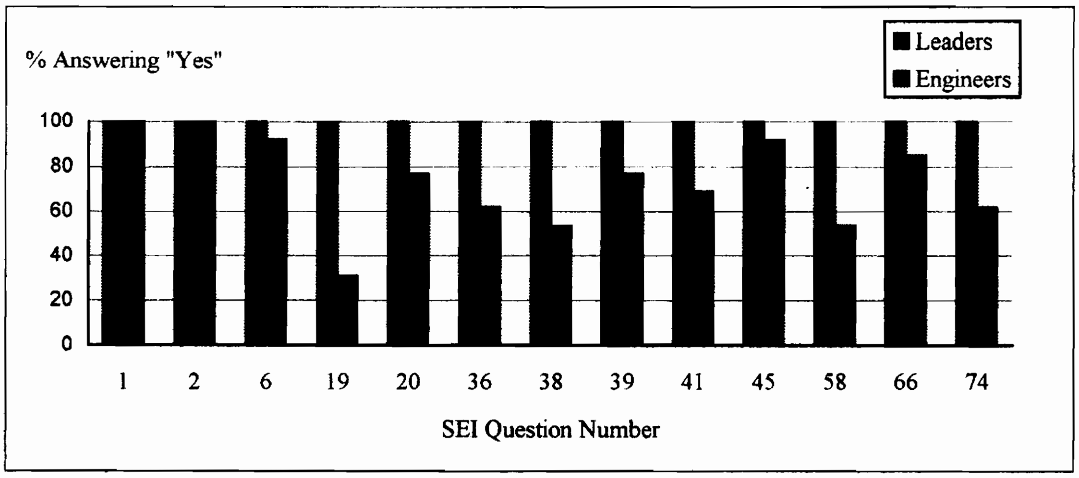


Figure 5-2. Level 2 questions to which all the leaders answer "yes".

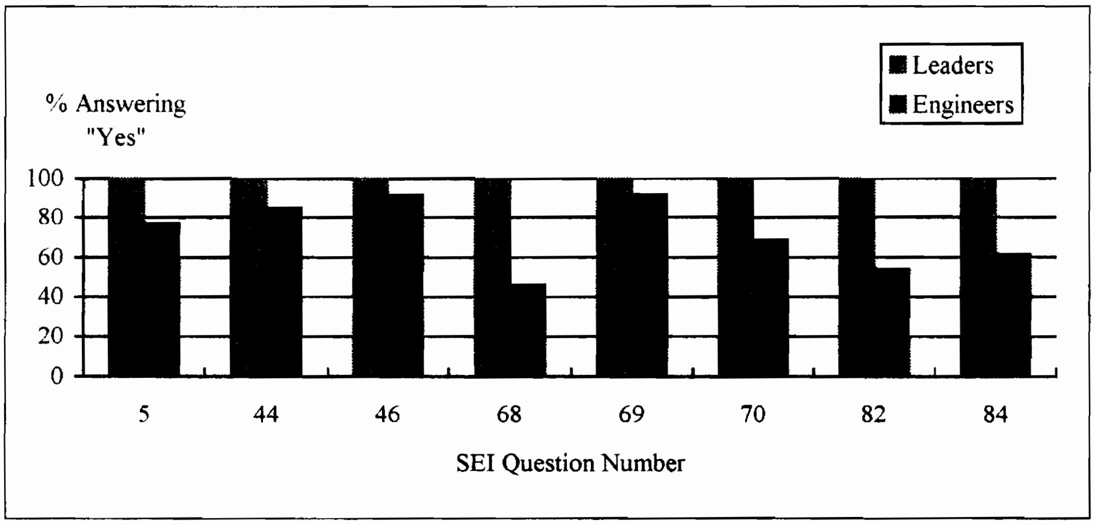


Figure 5-3. Levels 3, 4, and 5 questions to which all leaders answer "yes".

The leaders provide the same answers to the 4 questions to which all the engineers agree. Two of these questions are answered "yes" and are shown in Table 5-3 below. The answers of these two questions indicate that the organization has a well defined chain of command. The other two questions where all employees agree to the answer "no" are shown in Table 5-4 below. These two questions indicate the lack of software engineering training provided at XYZ.

Table 5-3. Two questions to which all employees answer "yes".

Question #1: For each project involving software development, is there a designated software manager?
Question #2: Does the project software manager report directly to the project (or project development) manager?

Table 5-4. Two questions to which all employees answer "no".

Question #10: Is there a required software engineering training program for software developers?
Question #11: Is there a required software engineering training program for first-line supervisors of software developers?

Summary

The discrepancies that are occurred in the initial survey results and are mentioned in this chapter cloud the view of the software development process at XYZ. These discrepancies may result from the differences in interpreting of terminology in the questionnaire between the employees or from the misunderstanding of how the organization performs its software activities. To have a better understanding about the process at XYZ, a follow-up interview is necessary. The questions that are used in follow-up interviews to elaborate the interviews are then generated from the discrepancies with the intention to clarify these discrepancies. The follow-up questions are provided in Appendix B. The interviews with the selected employees are arranged and conducted. The next chapter describes the follow-up interview and its results.

Chapter 6. Follow-up Interview Analysis

Introduction

The follow-up interviews are conducted with several of XYZ's employees who are selected based on their responses to the initial survey and the position they hold in the organization. The follow-up interview has proved to be a successful procedure in helping to clarify the discrepancies found in the initial survey data. The information collected from the follow-up interviews is used to determine whether or not each of the items listed in Table 5.2 is performed at XYZ.

During the interview, if an interviewee indicates that an activity is performed at XYZ, he or she should be requested to specify details about that activity. For example, the question of whether or not the organization is conducting formal reviews for each stage of software development is confirmed by some interviewees. These interviewees should then be able to specifically describe these formal reviews, the people who are required to attend, and how the reviews are conducted. With this method, if more than half of the interviewees provide the same information about an activity, the discrepancy about that activity can be clarified.

It is interesting to note that with the exception of one or two interviewees who indicated that they were not familiar with a specific activity addressed in the questions, most interviewees gave the same answers to all questions. The followed items are the general answers to the follow-up questions. The last two items were not addressed by the follow-up questions but brought up during the interviews.

- XYZ follows the DOD standard to generate their documents
- XYZ performs formal reviews as part of its software development process
- XYZ currently does not maintain software development data electronically

- XYZ currently does not have a well-defined mechanism for technical exchange
- XYZ has a change control process that is strictly enforced
- XYZ has not yet employed computer tools to assist its software development activities.
- XYZ's organizational structure has mixed development and quality control
- The main activities at XYZ involve software reverse-engineering

The following section discusses these findings in details. It also provides several insights into the organizational structure and software development activities of XYZ.

Analysis

The information obtained through the follow-up interviews is discussed in this section.

1. Although the questions listed in Table 5-3 indicate that XYZ has a well-defined chain of command, its organizational structure is not optimum. As illustrated in Figure 3-1, some contractors and in-house engineers are reporting to either one or all project leaders. This can cause the leaders difficulty in managing the development work. In addition, the CMQA personnel are under the development managers of XYZ. This appointment may interfere with the assuring of product quality due to the conflict of interest between the development and the product quality control. There also appears to be too much integration of development and IV&V agents. This again may cause conflict in quality enforcement.
2. The main responsibilities of XYZ involve mainly software maintenance and little development. The engineers often maintain prototype programs or upgrade existing baselines. When a prototype arrives at XYZ, the engineers deliver the program to their supported ship. The responsibility of XYZ's engineers is to ensure that the prototype works correctly on the ship. Since the prototype is designed at a high level of its functionality, several details may not be implemented. The engineers must exercise the code and modify it as necessary to ensure that it performs the specified requirements correctly without failures.

   In addition, since the organization is supporting many different baselines of the programs, it often happens that many requirements from one baseline are the same as or very similar to those of another baseline. Therefore, the engineers frequently use a program from a previous baseline as a base to develop the program for the current baseline. Changes are made to the base program until it performs all the requirements of the current baseline. In recent years, the engineers have started to generate design documents from the modified code.

3. The organization is using the DOD standard DOD-STD-2167A as a guide to generate the documents for its software. The DOD-STD-2167A is a set of standards developed by the DOD in 1988 to establish uniform requirements for software development of the entire project life cycle. It is described in [BENE92] as follows:

   > This standard includes the standards for configuration control, specification practices, engineering management, technical reviews and audits for systems, equipment and computer software, defense system software quality program, and for DOD data item descriptions which cover all phases of software development, maintenance and the production of the user manuals.

   This set of standards is intended to remove any ambiguities and ensure completeness in the software and its development process. The engineers have been following this standard closely.

4. XYZ performs formal reviews in the software development process. These include the System Requirement Review (SRR), Software Specification Review (SSR), Preliminary Design Review (PDR), and Critical Design Review (CDR) for each project. According to the DOD-STD-2167A, these reviews are defined as followed:

   - SRR is the review that is conducted when the analysis of the system requirements ends. The SRR determines if the system requirements are complete and ready for system design.
   - SSR is the review to determine whether the software requirements are complete and correct and that the design can begin.
   - PDR is the review after the preliminary design is complete. This review determines whether the detailed design phase should start or further modification to the preliminary design is required.
   - CDR is the review that gives the developers permission to start coding. However, in the case of XYZ, the code has already been generated. Therefore, this point determines the completion of development and prepares for delivery.

Except for the SRR which is done internally, the other reviews require the presence of the sponsors from NAVSEA. These reviews are also referred to as programmatic reviews. The reviews are concerned more with schedule, cost, and administrative matters than with technical issues. The sponsors decide whether the development may continue. The people who must attend these reviews are the project leader, the project leader's manager , any involved engineers, the IV&V agent who is responsible for the project, and the CMQA personnel. Since technical issues are not intensively discussed in these reviews, the organization also performs reviews that are called technical PDRs and technical CDRs prior to meeting with the sponsors. This way, they can carefully review the technical aspects of the design and prepare for the programmatic reviews. All personnel mentioned above, with the exception of the sponsors, are required to attend the technical reviews. Occasionally, the project leader formally invites some technical experts from other organizations to obtain their opinion of the project. The project leader is responsible for conducting all the meetings. The leader specifies the schedule for the meetings in the Software Development Plan (SDP) which has been developed by the project leader and approved by the project's sponsor before the development starts. XYZ holds very few code walk-throughs, since they rarely develop new code. Engineers sometimes arrange very informal code reviews of their own code. They consist of the author and a peer and are neither announced nor recorded.

5. XYZ currently has only one database that is electronically maintained. This is the database of the software trouble reports (STRs). This database is maintained by CMQA personnel. All other data and information are maintained on paper. All reviews can generate action items, which are maintained as hard copy and informally analyzed by responsible personnel. The results of the reviews and testing are collected and maintained as hard copies by CMQA. Although test results are not electronically recorded, the engineers do look back at the areas where problems occurred during testing and attempt to make the test procedures more comprehensive. They also document how they perform the tests and provide this information to responsible IV&V agents for verification and validation.

6. XYZ currently does not have a well-defined mechanism for technical exchange. Although XYZ employees, including in-house employees and contractors, have established a good working relationship among themselves, they do not have a well-defined mechanism for technical exchange. There are very few situations where information such as status reports and formal review announcements are formally distributed in the form of memoranda. The engineers also record problems that are reported by the supported ships and maintain them in hard copy form. Beside those few cases, engineers often discuss technical issues informally via e-mail, over the phone, or in-person. No records of these discussions are maintained, though some of them resolve major software problems.
7. XYZ has a change control process that is strictly enforced. After the code has been baselined, any change to it must go through a proposal and approval process. The engineer who finds the problems or who wishes to make a change to the code must submit to the local change control board (LCCB) an STR or an engineering change proposal (ECP). The ECCB chairperson reviews and approves or rejects the change request. Under this policy, before approval is granted, the code must not be changed. If the requested change effects any external interfaces, the engineer who requests the change must obtain an approval from the sponsors. Since this change control process is strictly applied, it often slows down the development process.
8. XYZ has not yet employed many computer tools to assist its software development activities. However, the organization is currently investigating several different tools. XYZ possesses the following computer-aided software engineering (CASE) tools that can be used in the organization's software reverse-engineering process:
   - RTM is a CASE tool that generates a requirement tracing matrix.
   - Interleaf is a document generator.
   - AdaMat evaluates code for complexity, enforces standards, and generates a requirement tracing matrix.
   - McCabe provides many software metrics and presents them graphically.

- Cadre's Teamwork can incorporate the output from the tools listed above, and generate a well-formatted document. However, this tool is very complicated to learn.

All these CASE tools are currently installed on XYZ's computer systems, but none of its products have been implemented using these tools. The engineers occasionally use RFFlow to generate flow charts or structure diagrams which are then incorporated into the documents that are created using general-purpose word processors such as Microsoft Word. However, the use of RFFLow is optional and very limited. In summary, XYZ does not make effective use of their currently available computer tools to aid in its software development or reverse-engineering processes.

Summary

The information which is obtained from the follow-up interview clarified the discrepancies which were introduced with the initial survey data. This information is studied and analyzed carefully to provide an in-depth understanding of the software development process at XYZ.

The analysis of the initial and follow-up results finds that the organization is currently at maturity level 1 of the CMM model.

A set of recommendations is then produced based on a combination of the initial survey findings, the conclusions from the follow-up interviews which have substantially improved our understanding about the organization's software development process, and the software engineering principle. The recommendation set is presented in the next chapter.

Chapter 7. Recommendations

Introduction

XYZ is attempting to improve its software development process. It currently does not have many signs of an organization at a higher maturity level. Yet, it possesses positive characteristics that can strongly support its attempt to improve. This chapter provides the most crucial information from the study. It discusses the positive characteristics of the organization and outlines the recommendations. These recommendations are also the result of research into prescribed methods to improve the software process. The intention of this set of recommendations is to suggest ways that the organization can begin improving its software process.

Recommendations

Because the organization is currently at level 1 of the CMM, the recommendations provided in this report encompass the areas which SEI has identified as the key process areas for an organization to advance from level 1 to level 2. These areas involve basic management control, such as project planning, project management, configuration management, and software quality assurance. This section discusses the recommendations in detail.

1. Clearly documenting responsibilities and job descriptions. Due to the limited personnel resources, the management often assigns an individual to various concurrent projects. Such appointments result in a situation where each individual reports to more than one project leader. This situation easily makes individual work status difficult to assess and control. In addition, work assignments and responsibilities are not always well-defined, which prevents management from efficiently utilizing all available personnel resources. One way to alleviate this problem is to clearly define and document the job descriptions, responsibilities, and channels of communication. In addition to assisting the management in managing its resources, the clear documentation of everyone's responsibilities also helps the engineers to successfully exchange or request technical information.

2. Limiting the use of prototype code. The software development at XYZ mainly involves reverse-engineering, that is taking the code from prototypes or from previous projects and generating the design and documents from the code. The engineers deliver the prototype code to the customer and modify it until the program works correctly at the customer site. They also select a previously baselined program, which satisfies a set of requirements that are closest to the new requirements, and then modify the program as necessary to fit the new requirements. From the interviews, it appears that many documents and programs contain information that is not necessary. This problem occurs because larger portions of code are transferred from earlier projects than what is needed. Such products greatly complicate the maintenance phase, whose cost is greater than that of the other three phases in the development life cycle -- requirement, design, and coding -- combined. Thus, this situation needs the immediate attention of the management. Two recommendations are made to improve this situation.

The first recommendation is that the reuse of prototype code should be carefully considered, reduced, or even eliminated. During the prototype development, changes that are made to reflect user needs are often implemented in an uncontrolled way. This means the changes are made as things go wrong until they become right and there is no documentation of the changes. If the prototype code is used in the final product, such changes cause difficulty in maintaining the product. The reason for this is that uncontrolled changes can degrade the structure of the product and thus make subsequent changes progressively more difficult [SOMI92]. According to SEI, if the prototype is not written with the intention of including the code in the final product, it is better to throw away the prototype than to try to maintain the prototype code [SEI95]. The SEI states that initial planning for the prototype code to be used in the final product requires considerations of design records, service facilities, and user documentation, and these items must be available when delivering the prototype.

3. Use of configuration management in reverse-engineering work. The second recommendation to improve the situation discussed in item 2 is the use of configuration management in the work at XYZ. Often large sections of code are reused from other projects, when only a small functionality of that code is really necessary. Because Configuration Management does not track the reuse, and object-oriented programming is not in use, this practice increases the cost of maintenance. A configuration management (SCM) can help to improve XYZ's reverse-engineering work. There should be a person designated as a configuration management librarian to manage the use of the existing code and documents that are maintained by SCM. Upon developer request, this person should determine if reuse is possible and identify the existing items that fit the request. If modifications are required before an existing item can fill a need, SCM personnel must record the changes and ensure that the developers perform the changes as needed. When appropriate changes are made, the unnecessary code can be eliminated from the new products. By using this approach, the code and document reuse becomes more systematic and organized. This can prevent the engineers from wasting their time searching for the most suitable items. It also makes the software much easier to maintain. Figures 7-1 and 7-2 illustrate the points made in these two recommendations. Figure 7-1 depicts the current process of software development at XYZ. As shown, this process contains four phases.

- Obtain prototype: The XYZ engineers obtain the prototype without documents from another organization.
- Compare with requirements: In this phase, the engineers run the prototype to determine whether it satisfies the requirements. If it does not, they determine whether there are portions of code from previous projects that can be incorporated into the prototype.
- Generate delivered code: This phase consists of using the code from the prototype, using the code from the previous projects, and developing new code to produce the code for the current project.
- Generate design document without Case tools: The engineers then use the final code as a reference as they generate the design documents using their favorite word processor.

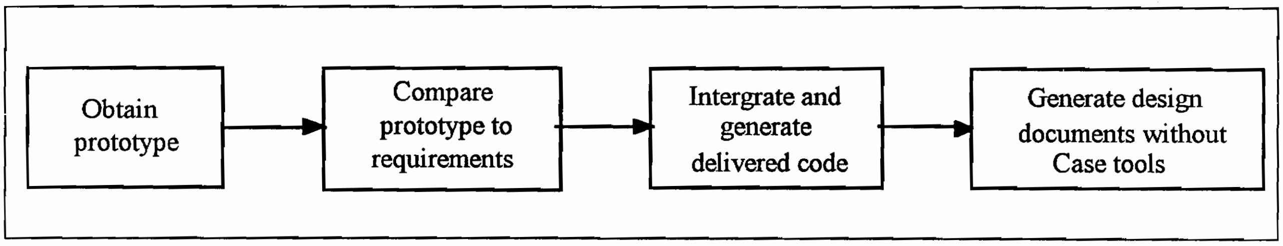

Obtain prototype
Compare prototype to requirements
Intergrate and generate delivered code
Generate design documents without Case tools

Figure 7-1. Current software development process at XYZ.

A recommended process to replace this current process at XYZ is provided in Figure 7-2. This recommended process includes five phases:

- Obtain prototype: The XYZ engineers obtain the prototype without documents from another organization.

- Evaluate the prototype: This phase consists of comparing the prototype against the requirements and identifying high-risk areas where redesign must be carefully considered. The comparison identifies missing requirements or implementations. If there are any capabilities that are in the prototype but not in the requirements, they must be discussed with the customer and added to the requirement document or eliminated from the product. If there are any requirements that are not accounted for in the prototype, they must be designed and implemented. The prototype helps the engineers to determine the best design before coding.
- Create design and generate design document using Case tools: In this phase, the engineers generate the design document from the complete requirement document. The generation of the design should be carried out with the help of their available CASE tools so that parts of the process can be automated. In addition, as recommended earlier, any document items that can be reused must be obtained from the CM librarian.
- Generate delivered code: At this phase, the engineers can use the complete design document to generate the delivered code. Again, code from previous projects can be obtained from the CM librarian.
- Integration and testing: Tests must be planned before the coding stage. These tests must be carried out thoroughly prior to delivery of the program to the ship.

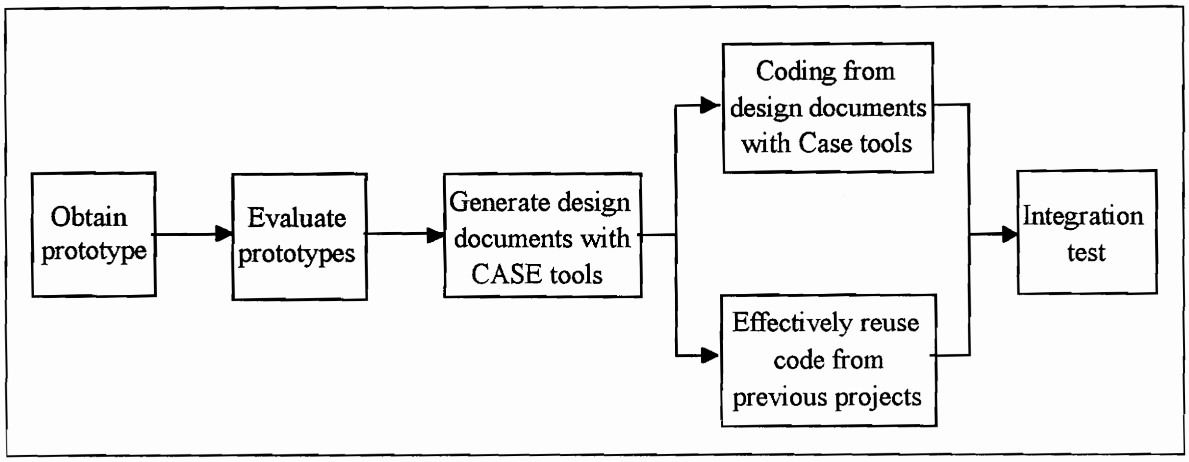

Obtain prototype
Evaluate prototypes
Generate design documents with CASE tools
Coding from design documents with Case tools
Effectively reuse code from previous projects
Integration test

Figure 7-2. Recommended process.

4. Providing software engineering training to the leaders. As shown in Table 5-4, there is no required software engineering training program for either software developers or first-line supervisors of software development. In order to have all employees participate in the process of improving the software development process, the management should first have the employees exposed to the software engineering literature. This way, all employees understand the value of improving the software process and their responsibilities in the improvement plan. Information about software engineering is widely available. Many good books about software development and management have been produced. Software engineers should be encouraged and supported to obtain this type of information. The existence of this material on the organization's bookshelves can also help to provide the professionals with an initial exposure to the software engineering area before actual training is available. At the least, the project leaders should immediately be provided with the required training on software management, specifically, on "how to conduct project reviews, what key indicators to examine, how to use basic management methods and tools, and methods for making quantitative quality estimates and plans" [HUMW89a][HUMW89b].

5. Tailoring the standards. Currently the organization is closely following the software development standard DOD-STD-2167A. This standard greatly contributes to the ease of managing very large projects but creates a very high overhead for small projects [BENE92]. The standard generates a large amount of documentation and therefore the time that is necessary to produce and maintain these documentation is also large. Small projects should not follow standards blindly, but should adapt the standards to the projects. In [BENE92], Bennatan makes this recommendation while referring to the requirement for tailoring specified in the foreword of DOD-STD-2167A:

> ... the standard must be appropriately tailored by the program manager to ensure that only cost-effective requirements are cited in defense solicitations and contracts.

The DOD has also provided a tailoring guide in the document DOD-HDBK-248. By adapting the DOD-STD-2167A standard to suit its projects, XYZ can save a large amount of resources that can be more effectively utilized in improving the development process.

6. Use available CASE tools more effectively. XYZ currently possesses several CASE tools that can contribute greatly to its software development process. The available tools include RTM, AdaMat, Interleaf, McCabe, and Cadre's Teamwork. However, these tools are not fully utilized. The engineers have the choice of whether or not they employ these tools in performing the software development, and they often choose not to use them. The reason for this seems to be the lack of knowledge about how to operate the tools. Some training about the tools has been provided, but enforcement may be necessary as well. When the engineers do not use a tool, they quickly forget any training that they have received. Thus, the training not being followed by practice becomes useless. It is recommended that the management enforce the use of the CASE tools in the software development. The initial steps of this enforcement can be using the available tools to generate documents and ensure standards.

7. Making training for new technologies a job assignment. XYZ has provided its employees with the opportunities to obtain technical training. However, some professionals devote most of their time to software development and neglect the need to learn new technologies. Training on new technologies keeps the professionals up-to-date and gives them insights into how these technologies can be applied to problems within the organization.

8. Establish a common mechanism for technical exchange. The initial survey results show that the employees are evenly divided on about half of the questions in the SEI questionnaire. Some of this disagreement may be considered as the result of misinterpretation of the questionnaire's terminology. However, the areas described in items 1, 4, and 5 in Table 5-2 seem to be the result of an organization-wide lack of communication. This kind of uncertainty among the employees about the organization's software development activities must be eliminated. Although the recommendation about clearly documenting individual responsibilities discussed in recommendation 1 can greatly reduce this undesirable situation, a mechanism for technical exchange is more crucial. Currently, the organization does not have a mechanism for professionals to exchange technical information. It is recommended that technical communications be recorded and maintained electronically. By maintaining this information, the engineers can go back to the discussion of problems and more quickly resolve similar problems. In addition, an answer recorded from a discussion about one project may be useful to another project and may save a developer time.

During the follow-up interview, some of the interviewees discuss a program called LotusNotes that the contractors are using at their site for technical communication. This program should be employed by the government employees as well. The use of this program not only helps XYZ provide its professionals a mechanism for technical exchange, but also gives the following benefits:

- The management can consult the contractors to determine the effectiveness of the program in this environment.
- The government employees and the contractors work very closely in developing the software at XYZ. Therefore, using the same program for communicating can help them keep their technical exchanges consistently recorded and easily tracked.

9. Establishing a Software Process Engineering Group (SPEG). A SPEG should be formed to assist the organization in making a plan for improvement and to ensure that all steps in the plan are performed correctly and consistently. According to Humphrey, management should assign about 2 percent of its professionals to the SEPG. However, if the available personnel resources do not allow that, at least one professional must be appointed to take this responsibility. Humphrey also identifies the characteristics that this professional must possess in order to be successful in this position:

- Enthusiasm about improving the software process. The following statement of Irwin and Langham explains this requirement [IRWH66]:

  > ...enthusiasm can be contagious and people tend to perform better in an optimistic environment than in a won't-work-environment.

- Ability to technically and politically understand the problems in the organization.
- Respectability by all professionals in the organization so that ensuring the implementation of improvement becomes smoother.
- Deserving the management's confidence and support so that his or her decisions and requests for improving actions are enforced by the management.
- Ability of making things happen so that the improvement process can be quickly accelerated.

Once such a person is selected, the management should ensure that he or she has some experience with software engineering, or otherwise, should provide him or her some special training on software engineering principle and practices. Being equipped with this knowledge, the SEPG personnel can help the organization in developing the improvement plan and in carrying out the plan effectively.

Summary

The assessment has provided insights into the software development process at XYZ. The information has been analyzed and used to generate several recommendations for process improvement. These recommendations are summarized below:

1. Document responsibilities and job descriptions
2. Limit the use of prototype code
3. Use configuration management for reverse-engineering work
4. Provide software engineering training to the leaders
5. Tailor the standards
6. Use the available CASE tools more effectively
7. Make training for new technologies a job assignment
8. Establish a common mechanism for technical exchange
9. Establish an SEPG.

These recommendations are all important in making improvement at XYZ. However, if the management needs to prioritize the recommendations and implement the highest-priority ones first, then a suggestion can be as followed.

Recommendation 4 needs to be implemented immediately because to effectively participate in organization-wide improvements, the employees need to know why the organization needs improvements and what every employee should do to realize the improvement plan and software engineering training can provide these concepts and practices.

The second highest-priority actions for the management to take are in recommendation numbers 1 and 8. Recommendation 1 urges the management to clearly document and publish individual responsibilities and recommendation 8 suggests the use of common tool for technical exchange. These two recommendations are intended to improve the effectiveness of technical communication within the organization. Effective communication is vital in a software organization.

Chapter 8. Conclusion

Introduction

This chapter summaries the study that is reported in this document. The next section provides a summary of the assessment. The section after that presents several conclusions reached through the study. These conclusions encompass the followings:

- The effect of the size of the studied organization on the study and on the process of improving the software development process.
- The importance of understanding, support, and willingness from the management.
- The positive attitude of the employees toward the assessment.

Each of these is briefly discussed in the conclusion section.

Study Summary

This study employs the CMM model developed by SEI to perform an assessment on the software development process at an organization which is identified as XYZ. In the study, several main activities have been carried out:

- Discussions with the management of XYZ to obtain permission to perform the assessment.
- Performing initial assessment using of the SEI questionnaire.
- Conducting follow-up interviews.
- Analyzing data collected from the initial assessment and the follow-up interviews.
- Generating recommendations for improvement.

In these activities, the analysis of the collected data and the generation of recommendations required the most time and consideration. The studied organization is small and limited in personnel resources. Tailoring the SEI software assessment principles, which are designed for large companies, to a small organization such as XYZ requires special consideration.

Conclusions of the Study

There are several conclusions that may be drawn from this study.

1. The size of the studied organization is relatively small - 17 members. The small size of the data set has the following effects on the study and the organization's improvement.
   - It is simpler to assess the process used in a small organization than that used in larger organizations.
   - Also the small organization is easier to control than a much larger organization. This helps the management to realize the improvement plan more quickly.
   - On the other hand, because of the limited number of personnel, it may be difficult for the organization to make plans for improvement. For example, forming an SEPG may require the management's significant consideration.

2. The support of the management is very crucial for the assessment process to be complete. They must allow the assessment to be performed and support it by providing the facility for surveys and requesting their employees' cooperation.

3. Most employees participating in the assessment are very enthusiastic. During the interviews, the interviewees tried to provide very thorough answers and explanations. This cooperative attitude greatly contributes to providing an outsider a better understanding of their software development process.

Summary

The effort of all those who participated in this assessment process should be used effectively and not discarded. Management should start making plans to implement the recommended changes. The earlier the actions are taken, the sooner the improvement will be realized.

It is a good practice to perform another assessment sometime after improvements have been made. The time between the assessments can be one year or more depending on several considerations such as budget and improvement schedule.

Last of all, the most important criteria needed for improvement is a positive attitude in the organization. If everyone is enthusiastic about improving the development process, it will improve.

Bibliography

[BOND95] Bond, D., "Project-Level Design Archetypes", *Software Development*, July 1995.

[CROP84] Crosby, P. B., "Quality Without Tears", McGraw-Hill, New York, 1984.

[DASM94] Daskalantonakis, M. K., "Achieving Higher SEI Levels", *IEEE Software*, July 1994.

[DIOR92] Dion, R., "Elements of a Process-Improvement Program", *IEEE Software*, July 1992.

[HUMW88] Humphrey, W. S., "Characterizing the Software Process: A Maturity Framework", *IEEE Software*, vol. 5, no. 2, March 1988, pp. 73-79.

[HUMW89a] Humphrey, W. S., "Managing the Software Process", Addison-Wesley, 1989.

[HUMW89b] Humphrey, W. S., Kitson, D. H., Kasse, T., "The State of Software Engineering Practice: A Preliminary Report", Software Engineering Institute, CMU/SEI-89-TR-1, DTIC Number ADA206573, February 1989.

[HUMW91] Humphrey, W. S., Snyder, T. R., and Willis, R. R., "Software Process Improvement at Hughes Aircraft", *IEEE Software*, vol. 8, no. 4, July 1991, pp. 11-23.

[IRWH66] Irwin, P. H., and Langham, F. W. Jr., "The Change Seekers", *Harvard Business Review,* January-February 1966, p.75.

[JENR79] Jensen, W. R., Tonies, C. C., "Software Engineering", Prentice Hall, 1979.

[JOHA94] Johnson, A. , "Software Process Improvement Experience in the DP/MIS Function", *IEEE*, 1994.

[SAIH95] Saiedian, H. and Kuzara, R., "SEI Capability Maturity Model's Impact on Contractors", *IEEE Computer*, January 1995.

[SEI95] Carnegie Mellon University, Software Engineering Institute, "The Capability Maturity Model: Guidelines for Improving the Software Process", Addison-Wesley, 1995.

[SOMI92] Sommerville, I. , "Software Engineering", Addison-Wesley, 1992.

Appendix A. Initial Questionnaire

The following section is to be filled out by the evaluator.

Organization :____
Name/Number :____

Evaluator's initials: ________________

SEI Assessment Questionnaire:

The following questionnaire was designed by software engineering researchers at SEI. Its sole purpose is to evaluate the software development process of the company. Individual results of this assessment will be kept confidential and will not be used to evaluate you.

There is no right or wrong in any of these questions, so do not worry if you ignore some of the terms used. However, try to answer all the questions.

Although this is not an individual evaluation, you are not supposed to discuss your answers with your colleagues until everybody is done.

PART I

Instructions: PART I

Please circle the option that best applies to you. In the question concerning your title/position, try to use standard terms (e.g., programmer, manager, CEO, and so on) for describing yourself.

1. How long have you been working in this company?

- 3 -- 6 years
- 1 -- 3 year(s)
- 6 mo. -- 1 year
- less than 6 mo.

2. What is your professional experience outside this company?

- more than 10 years
- 5 -- 10 years
- 1 -- 5 years
- less than 1 year

3 . What is your total professional experience as a software developer?

- more than 10 years
- 5 --10 years
- 1 -- 5 years
- less than 1 year

4 . What is your title/position in this company?

PART II

Instructions: PART II

Please answer **yes** or **no** to the following questions. Use the best of your knowledge, and answer **all** the questions. The person who gave you this questionnaire is not suppose to answer any questions concerning its contents. There is no time limit, but try to be as fast and accurate as you can. Thank you.

Questions: ***Please circle 'yes' or 'no' to each of the following.***

1. For each project involving software development, is there a designated software manager?	• YES • NO
2. Does the project software manager report directly to the project (or project development) manager?	• YES • NO
3. Does the Software Quality Assurance (SQA) function have a management reporting channel separate from the software development project management?	• YES • NO
4.. Is there a designated individual or team responsible for the control of software interfaces?	• YES • NO

5. Is software system engineering represented on the system design team?	• YES • NO
6. Is there a software configuration control function for each project that involves software development?	• YES • NO
7. Is there a software engineering process group function?	• YES • NO
8. Does each software developer have a private computer-supported workstation/terminal?	• YES • NO
9. Is there a required training program for all newly appointed development managers designed to familiarize them with software project management?	• YES • NO
10. Is there a required software engineering training program for software developers?	• YES • NO
11. Is there a required software engineering training program for first-line supervisors of software development?	• YES • NO
12 Is a mechanism used for maintaining awareness of the state of the art software engineering technology?	• YES • NO
13 Is a mechanism used for evaluating technologies used by the organization versus those externally available?	• YES • NO
14 Is a mechanism used for deciding when to insert new technology into the development process?	• YES • NO
15 Is a mechanism used for managing and supporting the introduction of new technologies?	• YES • NO

16. Is a mechanism used for identifying and replacing obsolete technologies?
• YES
• NO

17. Does the software organization use a standardized and documented software development process on each project?
• YES
• NO

18. Does the standard software development process documentation describe the use of tools and techniques?
• YES
• NO

19. Is a formal procedure used in the management review of each software development prior to making contractual commitments?
• YES
• NO

20. Is a formal procedure used to assure periodic management review of the status of each software development project?
• YES
• NO

21. Is there a mechanism for assuring that software subcontractors, if any, follow a disciplined software development process?
• YES
• NO

22. Are standards used for the content of software development files/folders?
• YES
• NO

23. For each project, are independent audits conducted for each step of the software development process?
• YES
• NO

24. Is a mechanism used for assessing existing designs and code for reuse in new application?
• YES
• NO

25. Are coding standards applied to each software development project?
• YES
• NO

26. Are standards applied to the preparation of unit test cases?
• YES
• NO

27. Are code maintainability standards applied?
• YES
• NO

28. Are internal design review standards applied?
• YES
• NO

29. Are code review standards applied?	• YES • NO
30. Is a formal procedure used to make estimates of software size?	• YES • NO
31. Is a formal procedure used to produce software development schedules?	• YES • NO
32. Are formal procedures applied to estimating software development cost?	• YES • NO
33. Is a mechanism used for ensuring that the software design teams understand each software requirement?	• YES • NO
34. Are man-machine interface standards applied to each appropriate software development project?	• YES • NO
35. Are software staffing profiles maintained of actual staffing versus planned staffing?	• YES • NO
36. Are profiles of software size maintained for each software configuration item, over time?	• YES • NO
37. Are profiles maintained of actual versus planned software units completing unit testing , over time?	• YES • NO
38. Are profiles maintained of actual versus planned software units integrated, over time?	• YES • NO
39. Are target computer memory utilization estimates and actuals tracked?	• YES • NO
40. Are target computer throughput utilization estimates and actuals tracked?	• YES • NO
41. Is target computer I/O channel utilization tracked?	• YES • NO

42. Are design and code review coverages measured and recorded?	• YES • NO
43. Is test coverage measured and recorded for each phase of functional testing?	• YES • NO
44. Are the action items resulting from design reviews tracked to closure?	• YES • NO
45. Are software trouble reports resulting from testing tracked to closure?	• YES • NO
46. Are the action items resulting from code reviews tracked to closure?	• YES • NO
47. Is test progress tracked by deliverable software component and compared to the plan?	• YES • NO
48. Are profiles maintained of software build/release content versus time?	• YES • NO
49. Has a managed and controlled process database been established for process metrics data across all projects?	• YES • NO
50. Are the review data gathered during design reviews analyzed?	• YES • NO
51. Is the error data from code reviews and tests analyzed to determine the likely distribution and characteristics of the errors remaining in the product?	• YES • NO
52. Are analyses of errors conducted to determine their process related causes?	• YES • NO
53. Is a mechanism used for error cause analysis?	• YES • NO

54. Are the error causes reviewed to determine the process changes required to prevent them?	• YES • NO
55. Is a mechanism used for initiating error prevention actions?	• YES • NO
56. Is review efficiency analyzed for each project?	• YES • NO
57. Is software productivity analyzed for major process steps?	• YES • NO
58. Does senior management have a *mechanism* for the regular review?	• YES • NO
59. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?	• YES • NO
60. Is a mechanism used for identifying and resolving system engineering issues that affect software?	• YES • NO
61. Is a mechanism used for independently calling integration and test issues to the attention of the project manager?	• YES • NO
62. Is a mechanism used for regular technical interchanges with the customer?	• YES • NO
63. Is a mechanism used for ensuring compliance with the software engineering standards?	• YES • NO
64. Do software development first-line managers sign off on their schedules and cost estimates?	• YES • NO
65. Is a mechanism used for ensuring traceability between the software requirements and top-level design?	• YES • NO
66. Is a mechanism used for controlling changes to the software requirements?	• YES • NO

67. Is there a formal management process for determining if the prototyping of software functions is an appropriate part of the design process?	• YES • NO
68. Is a mechanism used for ensuing traceability between the software top-level and detailed designs?	• YES • NO
69. Are internal software design reviews conducted?	• YES • NO
70. Is a mechanism used for controlling changes to the software design?	• YES • NO
71. Is a mechanism used for ensuring traceability between the software detailed design and the code?	• YES • NO
72. Are formal records maintained of unit (module) development process?	• YES • NO
73. Are software code reviews conducted?	• YES • NO
74. Is a mechanism used for controlling changes to the code? - Who can make changes and under which circumstances?	• YES • NO
75. Is a mechanism used for configuration management of the software tools used in the development process?	• YES • NO
76. Is a mechanism used for verifying that the samples examined by Software Quality Assurance are truly representative of the work performed?	• YES • NO
77. Is there a mechanism for assuring that regression testing is routinely performed?	• YES • NO
78. Is there a mechanism for assuring the adequacy of regression testing?	• YES • NO

79. Are formal test case reviews conducted?	• YES • NO
80. Is automated configuration control used to control and track change activity throughout the software development process?	• YES • NO
81. Are computer tools used to assist in tracing software requirements to software design?	• YES • NO
82. Are formal design notations such as PDL used in program design?	• YES • NO
83. Are computer tools used to assist in tracing the software design to the code?	• YES • NO
84. Is the majority of product development implemented in a high-order language?	• YES • NO
85. Are automated test input data generators used for testing?	• YES • NO
86. Are computer tools used to measure test coverage?	• YES • NO
87. Are computer tools used to track every required function and assure that it is tested / verified?	• YES • NO
88. Are automated tools used to analyze the size and change activity in software components?	• YES • NO
89. Are automated tools used to analyze cross references between modules?	• YES • NO
90. Are interactive source-level debuggers used?	• YES • NO
91. Are the software development and maintenance personnel provided with interactive documentation facilities?	• YES • NO

92. Are computer tools used for tracking and reporting the status of the software in the software development library?	• YES • NO
93. Are prototyping methods used in designing the critical performance elements of the software?	• YES • NO
94. Are prototyping methods used in designing the critical elements of the man-machine interface?	• YES • NO

Appendix B. Follow-up Questions

1. What kind of standards does the organization have in software development process? Which ones are actually followed? How closely are they followed? Who is responsible for the enforcement?
2. Are there formal reviews for each stage of the software development process? What are these reviews? Who are required to attend each of these reviews? How are they conducted and by whom?
3. Are the review data gathered during reviews in the software development process analyzed in depth? Are erroneous data from code reviews and tests recorded and reviewed to prevent reoccurrence? How are these data maintained?
4. What is the mechanism used for technical interchanges between engineers, between engineers and management, between in-house and contractors, and between in-house and customers? Is the mechanism well-defined and effective? Does it cause any discrepancies in the information exchange process?
5. How are changes to code controlled? How is the configuration of software tools managed? Is there a standard procedure to carry out a code change? How well is the procedure followed? Who can make changes and under which circumstances?
6. Are there computer tools used in tracing the software requirements to the software design and the software design to the code? What kind of tools? How much are they applied? What are other computer tools used in the software development process?

Appendix C. Key Practices

Level 2

1. Requirements Management

- The project follows a written organizational policy for managing the system requirements allocated to software.
- For each project, responsibility is established for analyzing the system requirements and allocating them to hardware, software, and other system components.
- The allocated requirements are documented.
- Adequate resources and funding are provided for managing the allocated requirements.
- Members of the software engineering group and other software-related groups are trained to perform their requirements management activities.
- The software engineering group reviews the allocated requirements before they are incorporated into the software project.
- The software engineering group uses the allocated requirements as the basis for software plan, work products, and activities.
- Changes to the allocated requirements are reviewed and incorporated into the software project.
- Measurements are made and used to determine the status of the activities for managing the allocated requirements.
- The activities for managing the allocated requirements are reviewed with senior management on a periodic basis.
- The activities for managing the allocated requirements are reviewed with the project manager on both a periodic and event-driven basis.
- The software quality assurance group reviews and/or audits the activities and work products for managing the allocated requirements and reports the results.

2. Software Project Planning

- A project software manager is designated to be responsible for negotiation commitments and developing the project's software development plan.
- The project follows a written organizational policy for planning a software project.
- A documented and approved statement of work exists for the software project.
- Responsibilities for developing the software development plan are assigned.
- Adequate resources and funding are provided for planning the software project.
- The software managers, software engineers, and other individuals involved in the software project planning are trained in the software estimating and planning procedures applicable to their areas of responsibility.
- The software engineering group participates on the project proposal team.
- Software project planning is initiated in the early stages of, and in parallel with, the overall project planning. The software engineering group participates with other affected groups in the overall project planning throughout the project's life.
- Software project commitments made to individuals and groups external to the organization are reviewed with senior management according to documented procedures.
- A software life cycle with predefined stages of manageable size is identified or defined.
- The project's software development plan is developed according to a documented procedure.
- The plan for the software project is documented.
- Software work products that are needed to establish and maintain control of the software project are identified.
- Estimates for the size of the software work products (or changes to the size of software work products) are derived according to a documented procedure.
- Estimates for the software project's effort and costs are derived according to a documented procedure.
- Estimates for the project's critical computer resources are derived according to a documented procedure.
- The project's software schedule is derived according to a documented procedure.
- The software risks associated with the cost, resource, schedule, and technical aspects of the project are identified, assessed, and documented.

- Plans for the project's software engineering facilities and support tools are prepared.
- Software planning data are recorded.
- Measurements are made and used to determine the status of the software planning activities.
- The activities for software project planning are reviewed with senior management on a periodic basis.
- The activities for software project planning are reviewed with the project manager on both a periodic and event-driven basis.
- The software quality assurance group reviews and/or audits the activities and work products for software project planning and reports the results.

3. Software Project Tracking and Oversight

- A project software manager is designated to be responsible for the project's software activities and results.
- The project follows a written organizational policy for managing the software project.
- A software development plan for the software project is documented and approved.
- The project software manager explicitly assigns responsibility for software work products and activities.
- Adequate resources and funding are provided for tracking the software project.
- The software managers are trained in managing the technical and personnel aspects of the software project.
- First-line software managers receive orientation in the technical aspects of the software project.
- A documented software development plan is used for tracking the software activities and communicating status.

4. Software Subcontract Management

- The project follows a written organizational policy for managing the software subcontract.
- A subcontract manager is designated to be responsible for establishing and managing the software subcontract.
- Adequate resources and funding are provided for selecting the software subcontractor and managing the subcontract.

- Software managers and other individuals who are involved in establishing and managing the software subcontract are trained to perform these activities.
- Software managers and other individuals who are involved in managing the software subcontract receive orientation in the technical aspects of the subcontract.
- The work to be subcontracted is defined and planned according to a documented procedure.
- The software subcontractor is selected, based on an evaluation of the subcontract bidders' ability to perform the work, according to a documented procedure.
- The contractual agreement between the prime contractor and the software subcontractor is used as the basis for managing the subcontract.
- A documented subcontractor's software development plan is reviewed and approved by the prime contractor.
- A documented and approved subcontractor's software development plan is used for tracking the software activities and communicating status.
- Changes to the software subcontractor's statement of work, subcontract terms and conditions, and other commitments are resolved according to a documented procedure.
- The prime contractor's management conducts periodic status/coordination reviews with the software subcontractor's management.
- Periodic technical reviews and interchanges are held with the software subcontractor.
- Formal reviews to address the subcontractor's software engineering accomplishments and results are conducted at selected milestones according to a documented procedure.
- The prime contractor's software quality assurance group monitors the subcontractor's software quality assurance activities according to a documented procedure.
- The prime contractor's software configuration management group monitors the subcontractor's activities for software configuration management according to a documented procedure.
- The prime contractor conducts acceptance testing as part of the delivery of the subcontractor's software products according to a documented procedure.
- The software subcontractor's performance is evaluated on a periodic basis, and the evaluation is reviewed with the subcontractor.
- Measurements are made and used to determine the status of the activities for managing the software subcontract.

- The activities for managing the software subcontract are reviewed with senior management on a periodic basis.
- The activities for managing the software subcontract are reviewed with the project manager on both a periodic and event-driven basis.
- The software quality assurance group reviews and/or audits the activities and work products for managing the software subcontract and reports the results.

5. Software Quality Assurance

- The project follows a written organizational policy for implementing software quality assurance (SQA).
- A group that is responsible for coordinating and implementing SQA for the project (i.e., the SQA group) exists.
- Adequate resources and funding are provided for performing the SQA activities.
- Members of the SQA group are trained to perform their SQA activities.
- The members of the software project receive orientation on the role, responsibilities, authority, and value of the SQA group.
- A SQA plan is prepared for the software project according to a documented procedure.
- The SQA group's activities are performed in accordance with the SQA plan.
- The SQA group participates in the preparation and review of the project's software development plan, standards, and procedures.
- The SQA group reviews the software engineering activities to verify compliance.
- The SQA group audits designated software work products to verify compliance.
- The SQA group periodically reports the results of its activities to the software engineering group.
- Deviations identified in the software activities and software work products are documented and handled according to a documented procedure.
- The SQA group conducts periodic reviews of its activities and findings with the customer's SQA personnel, as appropriate.
- Measurements are made and used to determine the cost and schedule status of the SQA activities.
- The SQA activities are reviewed with senior management on a periodic basis.

- The SQA activities are reviewed with the project manager on both a periodic and event-driven basis.
- Experts independent of the SQA group periodically review the activities and software work products of the project's SQA group.

6. <u>Software Configuration Management</u>
   - The project follows a written organizational policy for implementing software configuration management (SCM).
   - A board having the authority for managing the project's software baselines (i.e., a software configuration control board - SCCB) exists or is established.
   - A group that is responsible for coordinating and implementing SCM for the project (i.e., the SCM group) exists.
   - Adequate resources and funding are provided for performing the SCM activities.
   - Members of the SCM group are trained in the objectives, procedures, and methods for performing their SCM activities.
   - Members of the software engineering group and other software-related groups are trained to perform their SCM activities.
   - A documented and approved SCM plan is used as the basis for performing the SCM activities.
   - A configuration management library system is established as a repository for the software baselines.
   - The software work products to be placed under configuration management are identified.
   - Change requests and problem reports for all configuration items/units are initiated, recorded, reviewed, approved, and tracked according to a documented procedure.
   - Changes to baselines are controlled according to a documented procedure.
   - Products from the software baseline library are created and their release is controlled according to a documented procedure.
   - The status of configuration items/units is recorded according to a documented procedure.
   - Standard reports documenting the SCM activities and the contents of the software baseline are developed and made available to affected groups and individuals.
   - Software baseline audits are conducted according to a documented procedure.
   - Measurements are made and used to determine the status of the SCM activities.

- The SCM activities are reviewed with senior management on a periodic basis.
- The SCM activities are reviewed with the project manager on both a periodic and event-driven basis.
- The SCM group periodically audits software baselines to verify that they conform to the documentation that defines them.
- The software quality assurance group reviews and/or audits the activities and work products for SCM and reports the results.

Level 3

1. <u>Organization Process Focus</u>
   - The organization follows a written organizational policy for coordinating software process development and improvement activities across the organization.
   - Senior management sponsors and oversees the organization's activities for software process development and improvement.
   - A group that is responsible for the organization's software process activities exists.
   - Adequate resources and funding are provided for the organization's software process activities.
   - Members of the group responsible for the organization's software process activities receive required training to perform these activities.
   - Members of the SE group and other software-related group receive orientation on the organization's software process activities and their roles in those activities.
   - The software process is assessed periodically, and action plans are developed to address the assessment findings.
   - The organization develops and maintains a plan for its software process development and improvement activities.
   - The organization's and projects' activities for developing and improving their software processes are coordinated at the organization level.
   - The use of the organization's software process database is coordinated at the organizational level.
   - New processes, methods, and tools in limited use in the organization are monitored, evaluated, and, where appropriate, transferred to other parts of the organization.

- Training for the organization's and projects' software processes is coordinated across the organization.
- The groups involved in implementing the software processes are informed of the organization's and projects' activities for software process development and improvement.
- Measurements are made and used to determine the status of the organization's process development and improvement activities.
- The activities for software process development and improvement are reviewed with senior management on a periodic basis.

2. <u>Organization Process Definition</u>

- The organization follows a written policy for developing and maintaining a standard software process and related process assets.
- Adequate resources and funding are provided for developing and maintaining the organization's standard software process and related process assets.
- The individuals who develop and maintain the organization's standard software process and related process assets receive required training to perform these activities.
- The organizations standard software process is developed and maintained according to a documented procedure.
- The organization's standard software process is documented according to established organization standards.
- Descriptions of software life cycles that are approved for use by the projects are documented and maintained.
- Guidelines and criteria for the projects' tailoring of the organization's standard software process are developed and maintained.
- The organization's software process database is established and maintained.
- A library of software process-related documentation is established and maintained.
- Measurements are made and used to determine the status of the organization's process definition activities.
- The software quality assurance group reviews and/or audits the organization's activities and work products for developing and maintaining the organization's standard software process and related process assets and reports the results.

3. Training Program

- The organization follows a written policy for meeting its training needs.
- A group responsible for fulfilling the training needs of the organization exists.
- Adequate resources and funding are provided for implementing the training program.
- Members of the training group have the necessary skills and knowledge to perform their training activities.
- Each software project develops and maintains a training plan that specifies its training needs.
- The organization's training plan is developed and revised according to a documented procedure.
- The training for the organization is performed in accordance with the organization's training plan.
- Training courses prepared at the organization level are developed and maintained according to organization standards.
- A waiver procedure for required training is established and used to determine whether individuals already possess the knowledge and skills required to perform in their designated roles.
- Measurements are made and used to determine the status of the training program activities.
- The training program activities are reviewed with senior management on a periodic basis.
- The training program is independently evaluated on a periodic basis for consistency with, and relevance to, the organization's needs.
- The training program activities and work products are reviewed and/or audited and the results are reported.

4. Integrated Software Management

- The project follows a written organizational policy requiring that the software project be planned and managed using the organization's standard software process and related process assets.
- Adequate resources and funding are provided for managing the software project using the project's defined software process.
- The individuals responsible for developing the project's defined software process receive required training in how to tailor the organization's standard software process and use the related process assets.

- The software managers receive required training in managing the technical, administrative, and personnel aspects of the software project based on the project's defined software process.
- Each project's defined software process is developed by tailoring the organization's standard software process and is revised according to a documented procedure.
- The project's software development plan, which describes the use of its defined software process, is developed and revised according to a documented procedure.
- The software project is managed in accordance with the project's defined s/w process.
- The organization's s/w process database is used for software planning and estimating.
- The size of the s/w work products is managed according to a documented procedure.
- The project's s/w effort and costs are managed according to a documented procedure.
- The project's critical computer resources are managed according to a documented procedure.
- The critical dependencies and critical paths of the project's software schedule are managed according to a documented procedure.
- The project's software risks are identified, assessed, documented, and managed according to a documented procedure.
- Reviews of the software project are periodically performed to determine the actions needed to bring the software project's performance and results in line with the current and projected needs of the business, customer, and end users, as appropriate.
- Measurements are made and used to determine the effectiveness of the integrated software management activities.
- The activities for managing the s/w project are reviewed with senior management on a periodic basis and with the project manager on both a periodic and event-driven basis.
- The software quality assurance group reviews and/or audits the activities and work products for managing the software project and reports the results.

5. Software Product Engineering

- The project follows a written organizational policy for performing the software engineering activities.
- Adequate resources and funding are provided for performing the s/w engineering tasks.
- Members of the software engineering technical staff receive required training to perform their technical assignments and receive orientation in related software engineering disciplines.

- The project manager and all software managers receive orientation in the technical aspects of the software project.
- Appropriate software engineering methods and tools are integrated into the project's define software process.
- The software requirements are developed, maintained, documented, and verified by systematically analyzing the allocated requirements according to the project's defined software process.
- The software design is developed, maintained, documented, and verified, according to the project's defined software process, to accommodate the software requirements and to form the framework for coding.
- The software code is developed, maintained, documented, and verified, according to the project's defined software process, to implement the software requirements and software design.
- Software testing is performed according to the project's defined software process.
- Integration testing of the software is planned and performed according to the project's defined software process.
- System and acceptance testing of the software are planned and performed to demonstrate that the software satisfies its requirements.
- The documentation that will be used to operate and maintain the software is developed and maintained according to the project's defined software process.
- Data on defects identified in peer reviews and testing are collected and analyzed according to the project's defined software process.
- Consistency is maintained across software work products, including the software plans, process descriptions, allocated requirements, software requirements, software design, code, test plans, and test procedures.
- Measurements are made and used to determine the functionality and quality of the software products and the status of the software product engineering activities.
- The activities for s/w product engineering are reviewed with senior management on a periodic basis and with the project manager on both a periodic and event-driven basis.
- The software quality assurance group reviews and/or audits the activities and work products for software product engineering and reports the results.

6. <u>Intergroup Coordination</u>

- The project follows a written organizational policy for establishing interdisciplinary engineering teams.
- Adequate resources and funding are provided for coordinating the software engineering activities with other engineering groups.
- The support tools used by the different engineering groups are compatible to enable effective communication and coordination
- All managers in the organization receive required training in teamwork.
- All task leaders in each engineering group receive orientation in the processes, methods, and standards used by the other engineering groups.
- The members of the engineering groups receive orientation in working as a team.
- The software engineering group and the other engineering groups participate with the customer and end users, as appropriate, to establish the system requirements.
- Representatives of the project's software engineering group work with representatives of the other engineering groups to monitor and coordinate technical activities and resolve technical issues.
- A documented plan is used to communicate intergroup commitments and to coordinate and track the work performed.
- Critical dependencies between engineering groups are identified, negotiated, and tracked according to a documented procedure.
- Work products produced as input to other engineering groups are reviewed by representatives of the receiving groups to ensure that the work products meet their needs.
- Intergroup issues not resolvable by the individual representatives of the project engineering groups are handled according to a documented procedure.
- Representatives of the project engineering groups conduct periodic technical reviews and interchanges.
- Measurements are made and used to determine the status of the intergroup coordination activities.
- The activities for intergroup coordination are reviewed with senior management on a periodic basis and with the project manager on both a periodic and event-driven basis.

- The software quality assurance group reviews and/or audits the activities and work products for intergroup coordination and reports the results.

7. Peer Reviews
    - The project follows a written organizational policy for performing peer reviews
    - Adequate resources and funding are provided for performing peer reviews on each software work product to be reviewed.
    - Peer review leaders receive required training in how to lead peer reviews.
    - Reviewers who participate in peer reviews receive required training in the objectives, principles, and methods of peer reviews.
    - Peer reviews are planned, and the plans are documented.
    - Peer reviews are performed according to a documented procedure.
    - Data on the conduct and results of peer reviews are recorded.
    - Measurements are made and used to determine the status of the peer reviews activities.
    - The software quality assurance group reviews and/or audits the activities and work products for peer reviews and report the results.

Level 4

---

1. Quantitative Process Management
    - The project follows a written organizational policy for measuring and quantitatively controlling the performance of the project's defined software process
    - The organization follows a written policy for analyzing the process capability of the organization's standard software process
    - A group that is responsible for coordinating the quantitative process management activities for the organization exists.
    - Adequate resources and funding are provided for the quantitative process management activities.
    - Support exists for collecting, recording, and analyzing data for selected process and product measurements.

- The individuals implementing or supporting quantitative process management receive required training to perform these activities.
- The members of the software engineering group and other software-related groups receive orientation on the goals and value of quantitative process management.
- The software project's plan for quantitative process management is developed according to a documented procedure
- The software project's quantitative process management activities are performed in accordance with the project's quantitative process management plan.
- The strategy for the data collection and the quantitative analyses to be performed are determined based on the project's define software process.
- The measurement data used to control the project's defined software process quantitatively are collected according to a documented procedure.
- The project's defined software process is analyzed and brought under quantitative control according to a documented procedure.
- Reports documenting the results of the software project's quantitative process management activities are prepared and distributed.
- The process capability baseline for the organization's standard software process is established and maintained according to a documented procedure.
- Measurements are made and used to determine the status of the activities for quantitative process management.
- The activities for quantitative process management are reviewed with senior management on a periodic basis and with the project manager on both a periodic and event-driven basis.
- The software quality assurance group reviews and/or audits the activities and work products for quantitative process management and reports the results.

2. <u>Software Quality Management</u>
   - The project follows a written organizational policy for managing software quality.
   - Adequate resources and funding are provided for the managing the quality of the software products.
   - The individuals implementing or supporting software quality management receive required training to perform these activities.

- The members of the software engineering group and other software-related groups receive required training in software quality management.
- The project's software quality plan is the basis for the project's activities for software quality management.
- The project's quantitative quality goals for the software products are defined, monitored, and revised throughout the software life cycle.
- The quality of the project's software products is measured, analyzed, and compared to the products' quantitative quality goals on an event-driven basis.
- The software project's quantitative quality goals for the products are allocated appropriately to the subcontractors delivering software products to the project.
- The activities for software quality management are reviewed with senior management on a periodic basis and with the project manager on both a periodic and event-driven basis.
- The software quality assurance reviews and/or audits the activities and work products for software quality management and reports the results.

Level 5

---

1. Defect Prevention
   - The project follows a written organizational policy for defect prevention activities.
   - An organization-level team to coordinate defect prevention activities exists.
   - A team to coordinate defect prevention activities for the software project exists.
   - Adequate resources and funding are provided for defect prevention activities at the project and organization levels.
   - Members of the software engineering group and other software-related groups receive required training to perform their defect prevention activities.
   - The software project develops and maintains a plan for its defect prevention activities.
   - At the beginning of a software task, the members of the team performing the task meet to prepare for the activities of that task and the related defect prevention activities.
   - Causal analysis meetings are conducted according to a documented procedure.
   - Each of the teams assigned to coordinate defect prevention activities meets on a periodic basis to review and coordinate implementation of action proposals from the causal analysis meetings.

- Defect prevention data are documented and tracked across the teams coordinating defect prevention activities.
- Revisions to the organization's standard software process resulting from defect prevention actions are incorporated according to a documented procedure.
- Revisions to the project's defined software process resulting from defect prevention actions are incorporated according to a documented procedure.
- Members of the software engineering group and software-related groups receive feedback on the status and results of the organization's and project's defect prevention activities on a periodic basis.
- Measurements are made and used to determine the status of the defect prevention activities.
- The organization's activities for defect prevention are reviewed with senior management on a periodic basis and with the project manager on both a periodic and event-driven basis.
- The software quality assurance group reviews and/or audits the activities and work products for defect prevention and reports the results.

2. Technology Change Management

- The project follows a written organizational policy for improving its technology capability.
- Senior management sponsors and oversees the organization's activities for technology change management.
- A group responsible for the organization's technology change management activities exists.
- Adequate resources and funding are provided to establish and staff a group responsible for the organization's technology change management activities.
- Support exists for collecting and analyzing data needed to evaluate technology changes.
- Appropriate data on the software processes and software work products are available to support analyses performed to evaluate and select technology changes.
- Members of the group responsible for the organization's technology change management activities receive required training to perform these activities.
- The organization develops and maintains a plan for technology change management.
- The group responsible for the organization's technology change management activities works with the software projects in identifying areas of technology change.
- Software managers and technical staff are kept informed of new technologies.

- The group responsible for the organization's technology change management systematically analyzes the organization's standard software process to identify areas that need or could benefit from new technology.
- Technologies are selected and acquired for the organization and software projects according to a documented procedure.
- Pilot efforts for improving technology are conducted, where appropriate, before a new technology is introduced into normal practice.
- Appropriate new technologies are incorporated into the organization's standard software process and into the project's defined software processes according to a documented procedure.
- Measurements are made and used to determine the status of the organization's activities for technology change management.
- The organization's activities for technology change management are reviewed with senior management on a periodic basis.
- The software quality assurance group reviews and/or audits the activities and work products for technology change management and reports the results.

3. Process Change Management

- The project follows a written organizational policy for implementing software process improvements.
- Senior management sponsors the organization's activities for software process improvements.
- Adequate resources and funding are provided for s/w process improvement activities.
- Software managers and senior management receive required training in software process improvement
- The managers and technical staff of the software engineering group and other software-related groups receive required training in software process improvements.
- A software process improvements program is established which empowers the members of the organization to improve the processes of the organization.
- The group responsible for the organization's software process activities coordinates the software process improvement activities.
- The organization develops and maintains a plan for software process improvement according to a documented procedure.

- Members of the organization actively participate in teams to develop software process improvements for assigned process areas.
- Where appropriate, the software process improvements are installed on a pilot basis to determine their benefits and effectiveness before they are introduced into normal practice.
- When the decision is made to transfer a software process improvement into normal practice, the improvement is implemented according to a documented procedure.
- Records of software process improvement activities are maintained.
- Software managers and technical staff receive feedback on the status and results of the software process improvement activities on a event-driven basis.
- Measurements are made and used to determine the status of software process improvement activities.
- The activities for software process improvement are reviewed with senior management on a periodic basis.
- The software quality assurance group reviews and/or audits the activities and work products for software process improvement and reports the results.

Vita

Binh-Minh Tran, as the name might indicate to some people, is a Vietnamese. She grew up in Vietnam, and at the age of 20, while working towards a Bachelor's degree in Architecture, her family obtained permission to leave Vietnam and emigrate to the United States. On October 22, 1981, she entered a new life in the United States. When she arrived, she knew only the words "Thank you". With the strong support of her mother, Binh-Minh was able to move on with her new life.

Binh-Minh earned her Bachelor of Science in Computer Science at North Carolina State University in December, 1989. She has been working in a software development group for the Department of Defense since 1991. During the first four years there, Binh-Minh took several courses that are required for the Master's degree offered by Virginia Tech. Some of these courses involve Software Engineering. In 1995, her employer provided her with the opportunity to complete her degree more quickly by allowing her to attend school full-time. At Virginia Tech, she became more interested in Software Engineering and additionally Software Assessment, and decided to work with Dr. Sallie Henry to complete the rest of the requirements for the Master's degree.