An Integrated Approach to Software Process Assessment

By
Joel Henry

Dissertation submitted to the Faculty of Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree
of
Doctor of Philosophy
in
Computer Science

APPROVED:

Sallie M. Henry, Chairperson

Dennis Kafura

H. Rex Hartson

Lance Matheson

Jeff Birch

February, 1993
Blacksburg, Virginia

An Integrated Approach to Software Process Assessment

By
Joel Henry
Committee Chairperson: Sallie M. Henry
Computer Science

Abstract

This dissertation describes a methodology for assessing the software process (both development and maintenance) used by an organization. The assessment methodology integrates the principles of Total Quality Management and the work of the Software Engineering Institute. The integrated assessment methodology results in a well-understood, well-documented, quantitatively evaluated software process. The methodology utilizes four steps: investigation, modeling, data collection, and analysis of both process content and process output. The integrated assessment methodology was implemented at a large commercial software organization over a two year period. Implementation results are presented and significant conclusions are discussed. Four areas for further research are also presented.

Acknowledgements

I would like to express my sincere thanks to the following people and groups without whose support, encouragement, and input this dissertation would not be possible:

My wife, Kelley, whose infinite patience with my many hours of work, both at home and on-site, whose efforts to raise our sons and run our household essentially alone for nearly four years, and whose love, support, and encouragement provided the will to go on when I very much wanted to give up, is appreciated more than she will ever know.

My parents, Mom and Dad, thank you for always having just the right mix of sympathy for the hardships I was suffering and encouragement to persevere. My success stems directly from both of you.

My mother-in-law and father-in-law, thank you for never showing anything but confidence in my ability to complete this degree, and for tolerating the full year I spent with you just when you thought you finally had the house to yourself.

Dr. Sallie M. Henry, my advisor and friend, who agreed to work with me under difficult circumstances, helped me through the technical, administrative, and political difficulties of finishing this dissertation, and supported my efforts over great distances. Completion of this work was not possible without your tremendous support.

Robert M. Blasewitz, my advisor and friend, who gave me the opportunity to work in the demanding world of commercial computing, who allowed me to work on tasks directly related to my dissertation, and challenged me to think in ways not found in textbooks.

Dr. Lance Matheson, who agreed to serve on my committee despite my earring and without knowledge of my background. This work reflects a great deal of time and effort on your part that will always serve as an example of how a faculty member is to involve himself in the work of graduate students.

Dr. Dennis Kafura, whose experience, expertise, and candid comments focused and directed my efforts throughout this research effort. You will always represent a level of intelligence and professionalism for me to strive toward.

Dr. Jeff Birch, whose expertise in the area of statistics saved me a great deal of time and effort, and whose timely input made the statistical section of this dissertation the most important and most respected.

Dr. H. Rex Hartson, whose keen insight into the major issues considered in this research effort, and whose candid advice regarding my academic career at Virginia Tech are greatly appreciated.

The entire staff in the Computer Program Projects Group at GE, whose support of my work and tolerance of my questions and requests made this dissertation possible.

My sons, Matthew and Colin, who do not understand why I couldn't play with them so many times, or the weeks spent away from them, or the tremendous cost they have paid not having my time, yet never once wavered in their love and admiration for me. Such treatment I am not sure I deserve. Thank you.

Table of Contents

List of Figures

List of Tables

Chapter I. Introduction

Introduction

The development and maintenance of software have yet to employ a body of scientific knowledge and a measurable set of engineering practices similar to those used by such disciplines as Electrical Engineering, Mechanical Engineering and Chemical Engineering. Software Engineering as a discipline is struggling through the early stages of technical development. While statements such as "The software crisis is dead" have been made [HUMW89b], a realistic view of the current state of the practice places Software Engineering early in the commercial stage of technical development, as shown in Figure 1.1. Software Engineering is a craft becoming commercial.

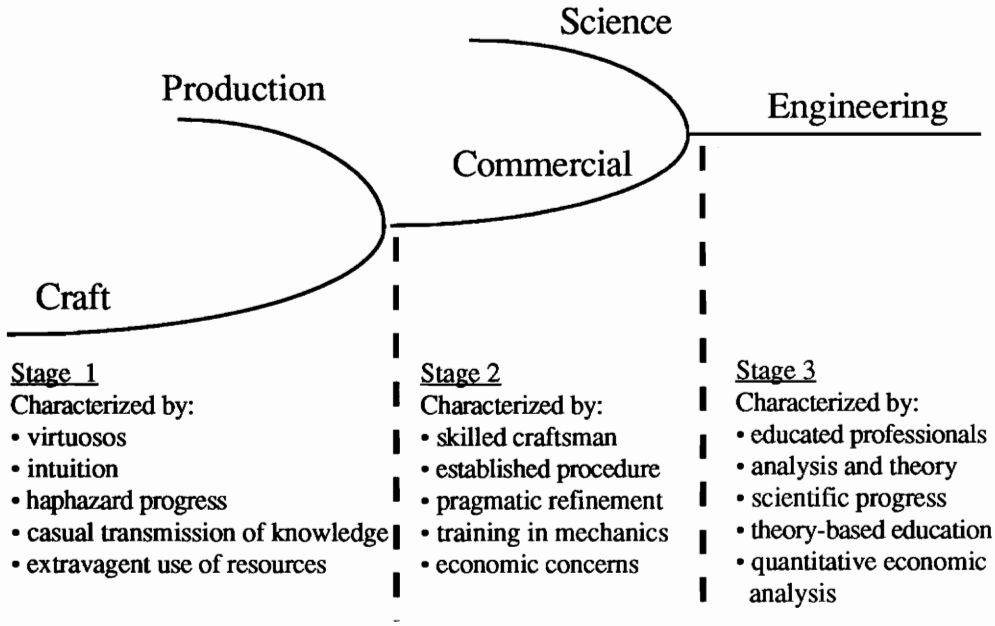

Figure 1.1 Stages of Technical Development

The need remains for the consistent production and effective maintenance of high quality software, on time and within budget. In a time of reduced budgets and increased competition for software contracts, organizations must improve the quality of the software products they develop and maintain while reducing cost.

The software process, including both development and maintenance, has significant impact on the quality, timeliness and cost of the software products produced [HUMW89b] [MOGJ90] [OSTL87] [BOLT91] [CRAS85]. Arthur et. al. substantiate a linkage of

process to product quality through the use of software engineering principles such as life cycle verification, early error detection and concurrent documentation [ARTJ86]. This work argues that the effective application of the appropriate software engineering principles imparts quality attributes to software products. The software process largely determines product cost and timeliness through software engineering management practices [QUIR80]. Huff et. al. have shown the management practices employed in the software process significantly impact product cost and delivery schedule [HUFK86].

Objectives

In order to focus this research effort, four objectives were defined:

1. To discover the activities comprising an organization's software process and the environmental factors affecting the process
2. To graphically document the process in a form usable by software personnel and amenable to analysis
3. To gather quantitative process and product data
4. To analyze an organization's process by considering activities present and absent within the process as well as statistical relationships between and among quantitative data

The first objective is to investigate the software process used by an organization. The history, mission, and environmental factors which impact an organization's process are also studied.

The second objective is aimed at documenting and clarifying the software process used by an organization. A graphic model with supporting written description is used to document an organization's software process.

The third objective requires process and product data to be specified, then collected during the software process and following delivery of the software product. The data is validated and stored in a central, automated database.

The fourth objective proposes software process analysis be performed in two ways. First, software process content is analyzed by considering the activities present and absent within an organization's software process. Second, the effectiveness of activities within the software process is analyzed by applying nonparametric statistical techniques to software process and product data.

Questions

The four research objectives are each quite broad and might be viewed as autonomous research efforts. This would be a serious flaw in any proposed software process assessment methodology.

In order to produce a cooperative, coherent software process assessment methodology the solution of each research objective must support all other solutions to the remaining three objectives. Given the objectives of this research and recognition of the need for each objective to support other objectives, the following questions must be asked, and solutions for each must be present in the proposed assessment methodology:

1. Can the activities within an organization's software process be discovered?
   Solution : The activities within an organization's software process can be discovered using an existing assessment method augmented with significant additional investigation techniques.
   This question relates to Objective 1. The solution techniques are presented in Chapter III and the results are discussed in Chapter IV.

2. Can the historical, environmental, and organizational factors impacting an organization's software process be determined?
   Solution : The historical, environmental, and organizational factors impacting an organization's software process can be determined by investigating these issues.
   This question relates to Objective 1. The solution techniques are presented in Chapter III and the results are discussed in Chapter IV.

3. Can an organization's software process be documented in a way usable to the organization and amenable to instrumentation?
   Solution : An organization's software process can be documented by applying control flow diagramming techniques to software process modeling.
   This question relates to Objective 2. The modeling technique is presented in Chapter III and the resulting model is presented in Chapter V.

4. Can an organization's process be instrumented to obtain accurate process and product data?
   Solution : An organization's software process can be instrumented by carefully considering the investigation results and the process model, and by precisely specifying the data to be captured.
   This question relates to Objective 3. The measurement techniques are presented in Chapter III and the data obtained is discussed in Chapter V.

5. Can the content of an organization's software process be effectively analyzed?
   Solution : The content of an organization's software process can be analyzed by comparing the organization's software process to established software engineering principles and software management practices.
   This question relates to Objective 4. Software process content analysis techniques are described in Chapter III and the resulting analysis is presented in Chapter V.

6. Can statistically valid relationships between and among process data and product data be discovered and used to analyze an organization's software process?
   Solution : Statistically valid relationships between and among process data and product data can be established using nonparametric statistical techniques.
   This question relates to Objective 4. The statistical techniques are described in Chapter III and the resulting analysis is presented in Chapter VI.

The succeeding section of this chapter discusses the software process in more detail. The remaining section of this chapter presents the two existing process assessment methods having the greatest influence on this research and a discussion of how these methods are integrated.

The Software Process

The software process can be divided into activities and subprocesses. Process activities are the actions taken by personnel to produce a product. Activities may be specified at a high level, such as "Construct an Automobile Body", or at a low level of detail, such as "Weld Joint # 34". Process activities are the production steps used to create a product, in the same way each task along an automobile assembly line is used to construct an automobile. A subprocess is a series of activities within a process with a common goal. Subprocesses are series of activities forming a process of their own. For example, the activities used to construct and finish the body of an automobile form a subprocess of automobile manufacturing.

In order to assess the software process, the purpose of each subprocess and activity must be understood. Subprocesses and activities are performed within the process to support a software engineering principle, a project management practice, or an environmental requirement (i.e. contractual, organizational, governmental). For example, an activity such as code walkthroughs, support the software engineering principle of early error detection [ARTJ86]. In-Process Reviews provide management with project status information [HUFK86]. The periodic review of project personnel security clearances satisfies Department of Defense requirements for project security.

Software Process Improvement

Assessing and improving the software process are important tasks. Process assessment results in a well-understood, well-documented and quantitatively evaluated software process. Only in a well-understood process can the impact of an improvement be fully measured. Process improvements are more easily implemented, and more permanent, when the process is documented. Quantitative evaluation of the process supports objective feedback on the benefit provided by process improvement.

The Software Engineering Institute (SEI) has been in the forefront of software process research during the past five years, producing a process maturity framework, a capability assessment procedure and a capability maturity model (currently under review, scheduled for release in early 1993) [HUMW87] [SEI91]. SEI efforts have drawn both praise and criticism [BOLT91] [HUMW91b] [MOGJ90]. While debate over the SEI approach continues, many experts in the field of Software Engineering agree that the greatest advancements in the field will result from assessing and improving the software process.

However, abundant evidence exists that a complete, coherent, validated methodology for process assessment and improvement is absent. For example, three software development organizations, Hughes, General Electric and Hewlett Packard, have undertaken significant, costly, long-term efforts to improve the software processes they use [HUMW91a] [GESD91a] [GRAR82]. Unfortunately, each of these three organizations have implemented different process improvement plans. While similarities exist between the plans, striking differences can also be found. Hewlett Packard implemented an ambitious improvement plan, focusing on product measurement. The Hughes improvement plan is based on the SEI maturity framework, requiring that specific activities be added to the development process to achieve improvement. Process improvement at General Electric utilizes concepts from the SEI work but also relies heavily on the principles of Total Quality Management.

Software Process Assessment

Software process assessment is the critical first step in software process improvement. Assessment provides information about what type of process exists, how effective the process is and where problem areas within the process exist. Process improvement not based on a thorough assessment is at best a collection of educated guesses, and the results obtained can only be subjectively judged as well.

The two most important approaches to software process assessment and improvement are the Software Engineering Institute (SEI) approach and the principles of Total Quality Management (TQM) [SEIS92]. The SEI Contractor Capability Assessment, Process Maturity Framework, and Capability Maturity Model represent significant advancement in the field of software process research. The principles of TQM, including the use of teams, application of statistical analysis methods, and focus on the customer, have proven successful in other industries.

SEI process assessment, based on investigation of the software process, establishes what activities and tools exist within the process. The SEI assessment method evaluates a process based on substantiated answers to 101 yes/no questions. The questionnaire results evaluate an organization's process on a process maturity framework containing five levels.

The process maturity levels within the Process Maturity Framework form a logical progression of more sophisticated software process descriptions. The activities included in each maturity level are drawn from accepted software engineering practices and proven industrial engineering techniques. Despite these strengths, serious questions exist regarding the grading scheme used in the assessment procedure [BOLT91]. The abscence of a technique to tailor the assessment method to the organization based on the historical and environmental pressures is also seen as a problem [HENJ91a].

TQM (Total Quality Management) is a general approach to product improvement focusing on the development process. TQM involves defining the process used by an organization, measuring both product and process, and using statistical techniques to analyze the process [DOD88]. Commitment to improvement throughout the organization and the use of teams are also important ingredients in a successful TQM effort.

The strengths of TQM are reliance on objective measurement, the application of statistical quality control, and the commitment of all personnel to quality throughout development [DOD88]. The application of TQM to software development is not clearly defined. The development activities used in a manufacturing environment are well understood (activities such as welding, soldering, painting, etc.) and involve physical components. Software process activities are not as well understood and operate on concepts and abstractions. Exactly what activities a given organization should include in its software process are not clearly defined.

SEI Software Process Assessment

SEI Contractor Capability Assessment satisfies the need to discover and understand problems within an organization's software process. The assessment is a balance between an unconstrained search for problems and the urge to prematurely specify solutions.

The objectives of SEI process assessment are to:

- learn how the organization works,
- identify problem areas within the process, and
- enroll it's leaders in the change process [HUMW89b].

The SEI Contractor Capability Assessment method is based on the five-level Process Maturity Framework [HUMW89b]. The process maturity levels are:

1. Initial - Ill-defined procedures and controls result in little consistency in managing the software process.

2. Repeatable - An organized and stable software process results in the ability to measure and trace the process and produce accurate estimates of costs and schedules.

3. Defined - A well-defined software process and the collection of tracking data allow verification of compliance with designated standards.

4. Managed - Analysis of quantitative measures of all aspects of the software process results in control of the process.

5. Optimized - Control of the software process provides an opportunity to modify and improve the process using process data.

The process maturity levels form a logical progression of more sophisticated process descriptions. The activities required in each level implement accepted software engineering practices and proven industrial engineering techniques.

SEI process assessments involve the following steps:

1. Secure management commitment to change prior to assessment.
2. Administer the questionnaire and collect the results.
3. Conduct follow-up interviews to substantiate the results of the questionnaire.
4. Give a preliminary presentation of the assessment results.
5. Produce a thorough written report of the assessment results.

The SEI assessment method begins by receiving management commitment to the assessment and subsequent improvement. Without such commitment assessment is difficult if not impossible and subsequent improvements have little chance of implementation.

The second and third steps gather substantiated answers to 101 yes/no questions [HUMW87]. The questions are separated into groups corresponding to process maturity levels 2 through 5. The purpose of each group is to evaluate the organization's process against the required activities for a specific maturity level.

Questionnaire and follow-up data assess the maturity of an organization's software process as having achieved one of the five levels in the Process Maturity Framework. The organization's process improves by adding the activities missing from an organization's process, but required by the SEI Process Maturity Level.

The SEI process assessment detects the presence or absence of the activities required to achieve a specific maturity level. However, the assessment method elicits much more information than is captured [BOLT91]. Activities not specified within the Process Maturity Framework are not detected. In addition, the assessment method does not specify how to tailor assessment results to an organization. Organizational goals and environmental factors are not considered in the associated assessment, consequently this type of information is not captured [HENJ91a].

TQM Process Assessment

TQM is a general approach to product quality improvement. TQM is "A cooperative way of doing business that relies on the talents and capabilities of both management and labor to continually improve quality and productivity using teams." [JABJ91] The six principles of TQM are as follows:

1. Maintain customer focus.
2. Focus on the process.
3. Prevent defects rather than inspect for defects.
4. Utilize the knowledge and expertise of labor.
5. Perform fact-based decision making.
6. Integrate feedback to continuously improve the process.

These general purpose principles are applicable to nearly all types of industry, and particularly to the commercial production of software. The emphasis on process assessment and improvement reflects the belief that quality products result from a quality process.

TQM includes five phases:

- preparation,
- planning,
- assessment,
- implementation, and
- diversification.

Organizational assessment is typically performed through self-evaluation, customer surveys or training feedback. No single structured approach to organizational assessment is specified in TQM.

The implementation phase requires process definition, measurement and improvement based on statistical analysis [DOD88]. Process definition documents the process and supports specification of measurement points within the process. Measurement quantifies product quality and process effectiveness at various points throughout the process. Control charts are established to monitor and manage the process. These control charts show the average number of product defects detected following each activity and control limits. The control limits depict deviations from the average based on the inherent variability of the process. If the number of product defects exceeds control limits the process is out of control and management alerted. Process improvement is achieved by reducing the average number of product defects. An example of a control chart is shown in Figure 1.2.

The control chart presented in Figure 1.2 depicts the number of defects detected in a manufactured part over time. The upper and lower control limits are typically one standard deviation above and below the average number of defects detected. The spike in the center of the chart reflects a number of defects above the upper control limit. Detected defects above the upper control limit prompt management to take corrective action to lower the number of defects to within the control limits. Detected defects within control limits represents product quality within accepted quality standards. Defect numbers below the lower control limit may also prompt management action.

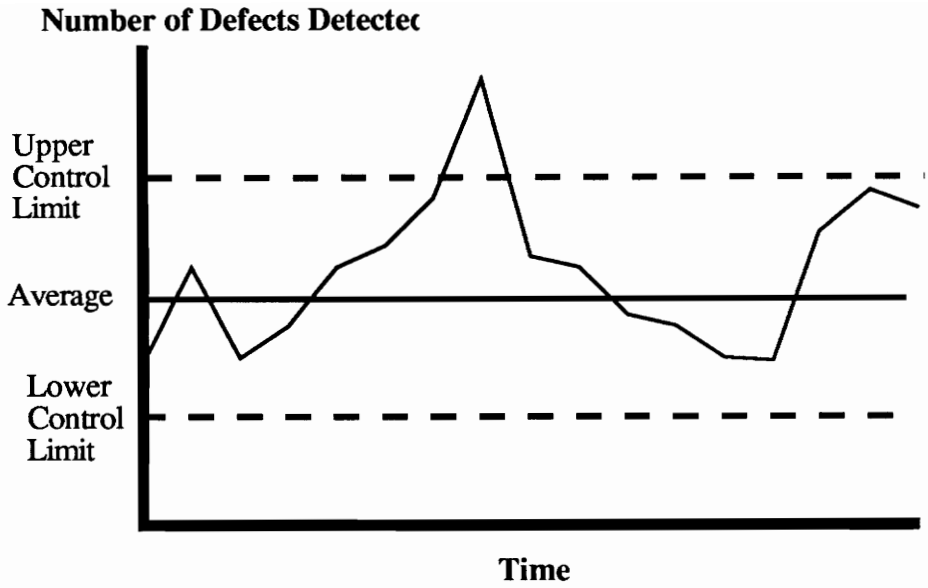

Figure 1.2 Control Chart

Integrating SEI and TQM Process Assessment

The SEI and TQM approaches differ significantly. The SEI approach can be viewed as a process content assessment while TQM can be viewed as a statistical data assessment. SEI process assessment examines activities within the software process in detail, adding activities or altering the flow of control between activities. TQM also considers the activities within the process but concentrates on improving product measures following each task.

The assessment methodology outlined in this dissertation integrates concepts from both the SEI approach and the TQM principles. The basis for integrating these two approaches is:

The SEI assessment specifies the activities comprising the software process.
The TQM implementation phase evaluates the effectiveness of the activities.

The investigation step of the methodology determines the software process used by an organization and the environmental factors affecting the process. Investigation is the basis for process documentation in the form of a process model. The process model and

investigation results support instrumentation of the software process. Quantitative process assessment utilizes statistical analysis techniques.

The methodology utilizes both the SEI approach and the TQM approach to process assessment. Investigation and modeling discover and document the activities used by an organization. Data gathering and statistical analysis discover product-process relationships throughout the software process. Consideration of the activities within the process and statistical relationships are needed to determine what types of improvements are required and how to implement the improvements.

Summary

This chapter introduced the concept of software process assessment and stressed the importance of assessment to an effective process improvement effort. The two most important existing approaches to process assessment were described. An approach to integrating them into a single coherent assessment methodology was presented.

Four objectives for this research effort were stated. In order to focus the research and achieve the objectives, six questions were posed and solutions advanced. Detailed descriptions of the solutions and the results of applying the solution techniques are presented in the following chapters.

Chapter II discusses previous work in the areas of process investigation, process modeling, data gathering, and process analysis. Chapter III describes the specific techniques used in the integrated process assessment methodology. Chapter IV describes the implementation of the integrated assessment methodology within the Government Electronic Systems Division of General Electric. Chapter V presents the results of the integrated assessment methodology, including achievement of Objectives 2, 3 and 4. The achievement of Objective 1 is detailed in [GESD91b] and is company proprietary. Chapter VI summarizes the research effort and presents conclusions. The direction of future work is also presented in Chapter VI.

Chapter II. Process Assessment

Introduction

Software process assessment provides information about the nature of the process, the effectiveness of the process, and the existence of problem areas within the process. The objectives of process assessment are to:

- learn how the organization software process works, and
- identify problem areas within the process [HUMW89b].

Process assessment satisfies the need to discover and understand problems within an organization's software process. An assessment is a balance between an unrestrained search for problems and the urge to prematurely specify solutions.

A thorough approach to software process assessment, intended to be used by an organization over a long time period (many years), must include investigation, modeling, measurement, and analysis. In order to understand the software process used by an organization, the process must be investigated. The process must be documented in a form usable to the organization and amenable to analysis for effective measurement and improvement to be performed. The process model and the quantitative measurement of both process and product must be used to perform a complete assessment of an organization's process.

This chapter discusses in detail previous process assessment methods and other techniques related to process assessment. The first section describes techniques used to investigate an organization's software process. The second section presents approaches to modeling software processes. Process and product measurement techniques are presented in the third section. Techniques for analyzing an organization's software process are described in the fourth section. The fifth section summarizes the material presented.

Process Investigation

Every software organization utilizes a software process to build and maintain computer programs and their associated documentation. The software process is often undocumented and frequently is misunderstood by the personnel using it [HENJ91a]. The purpose of process investigation is to discover the software process used by an organization and to understand the process in view of the environment in which the process operates. Environmental factors (such as equipment, budgets, schedules, manpower, and competition) affecting the organization are an important part of understanding the software process an organization employs.

Investigation of the software process answers the following questions:

- What activities make up the process?
- How do the activities interrelate?
- What methods and tools are used to perform the activities?
- How does organizational structure influence the process?
- How do environmental factors affect the process?

Previous work by Crawford and Fallah describe a process audit procedure using lengthy interviews [CRAS85]. The procedure captures organizational and environmental factors that influence the software process. In fact, a great deal of information is captured from their process audits. The information obtained is subjective and is not amenable to statistical tests. No process model or framework is used to guide the investigation. The procedure focuses on capturing information about organizational problems from the perspective of development personnel.

The SEI Contractor Capability Assessment Method is a question-based investigation structured by the Process Maturity Framework. While the evaluation method has been criticized [BOLT91] [HENJ91a], the techniques used to investigate the process are very effective at determining the presence or absence of the specific set of process activities required by the maturity framework.

The investigation portion of the SEI assessment method begins by administering a questionnaire containing 101 yes/no questions. The questions are separated into seven sections. The first three sections concentrate on organizational structure, support of personnel and availability of resources. The remaining four sections deal directly with the software engineering process and its management. Groups of questions from each section are used to evaluate the organization against the five-level maturity framework.

Results of the questionnaire are used to determine what type of questions are included within the follow-up interviews. The purpose of follow-up interviews is to substantiate the answers obtained from the questionnaires. Questionnaire and follow-up information is then used to assess the software process used by an organization, labeling the organizations process as maturity level 1, 2, 3, 4, or 5.

The SEI process investigation is successful in gathering data which can be statistically tested [HENJ91a]. The presence or absence of the activities required to achieve a specific maturity level is discovered. However, the SEI method does not detect all the activities used by an organization. Activities not specified within the Process Maturity Framework are not detected [BOLT91] [HENJ91a]. In addition the investigation method does not specify how to tailor investigation results to an organization. Organizational goals and environmental factors are not considered in the associated assessment, consequently this type of information is not captured. Bolten and McGowen assert that much more information is elicited than captured in the investigation method [BOLT91].

Effective process investigation results in a well-understood software process and provides the necessary information for documenting the software process. Objective and subjective information are used to determine when and where to instrument the process. The information obtained is also critical to determining what process improvements are needed and how to implement the improvements. Any attempt to document or improve the process without the benefit of information gained from investigation is done in some degree of darkness [CURB87].

Process Modeling

In order to assess and improve a software process, the process must be accurately documented. Process documentation illuminates the software process across the organization. Process documentation is critical in determining what type of data to collect, where to collect data and how to interpret the data. An important part of process documentation is a graphic model of the development process.

Process modeling has become an active research area in the recent past, as demonstrated by the abundance of manuscripts published in workshops and conferences during the last five years (i.e. the International Conference on Software Engineering, the International Software Process Workshop and the Hawaiian International Conference on System Sciences) [KELM89d]. Significant effort has been devoted to process modeling at the Software Engineering Institute [HANG88] [KELM88] [KELM89a] [KELM89b] [KELM89c] [KELM89d] [HUMW89a] [KELM90]. Process modeling is also a critical component in the application of Total Quality Management to software development.

Process models document the development process by:

- showing the activities comprising the software process,
- communicating the roles and responsibilities of both individuals and groups,
- specifying when software products are created and changed,
- depicting where measurement is performed,
- describing the communication channels between development groups.

The earliest process models constructed, such as the waterfall, spiral and evolutionary lifecycle models show the highest level tasks taking place during software development. The objective of most phases within these models is the production of a software product. For example, the initial three phases of the popular waterfall model specify the production of a requirements document, a design specification, and the executable code. The activities within each phase are virtually invisible in these models because of the emphasis on phase-ending products and lack of lower level process specification [CURB87].

Osterweil compares the specification of software processes to cookbook recipes, procedure manuals and assembly instructions [OSTL87]. Osterweil uses this comparison to argue that software processes are software and therefore can be defined using programming language constructs. While this approach to process documentation specifies detailed activities within each phase, the understandability, usability and applicability of programming language constructs to process representations is questionable.

More recent approaches to process modeling represent software processes from one or more of the following three perspectives:

1. Behavioral - The behavioral perspective represents when tasks are performed, and some aspects of how they are performed through feedback loops, iteration, decisions, entry and exit conditions, etc.

2. Functional - The functional perspective represents what tasks are being performed and what information flows are pertinent to those tasks.

3. Organizational - The organizational perspective represents where in the process, and by whom within the organization, the tasks are performed as well as the physical communication between individuals and groups.

The behavioral modeling approach presented by Williams integrates timing with a graphical representation of tasks [WILL88]. Information flow, critical in the development of large software products, is not shown in the models.

Process models created using the functional view capture information flow and task description. The software process descriptions proposed by Katayama are not graphical in nature and their use in the demanding world of commercial computing is suspect [KATT89a] [KATT89b].

Katamaya and Kellner integrate organizational information into the modeling method [KATT89a] [KATT89b] [KELM89]. Organizational information is critical in a large development environment and important to those responsible for auditing the software process; however this type of representation is not measurable.

Entity process models, overlapping behavioral and functional boundaries, have also been applied to process modeling [HUMW89]. The entity process model specifies entry and exit criteria as well as the task performed. This type of modeling is well suited to software processes because software processes are comprised of activities. However, entry and exit criteria are difficult to specify for many activities because process goals change from project to project [BASV87].

SEI process models represent behavioral, functional and organizational views of a process by creating three separate models, one for each view [KELM89d]. The presence of three models requires a software developer to consult all three models, then form an integrated representation of the process. The modeling technique allows multiple levels of detail on a single page, but the purpose of each level and the relationship between levels is not clearly specified.

Each of the process modeling approaches discussed here attempts to define and describe the software process. The recent level of activity in the area of process modeling shows the increasing focus on the software process as an important object of study. The emphasis of this focus is in assessing and improving the software process. Process models are a critical part of process assessment. Process models document what process activities exist, what types of data to collect, and where process data is collected.

Data Gathering

The more mature engineering disciplines, such as Electrical Engineering and Mechanical Engineering, integrate measurement of both the manufacturing process and product characteristics into the development of new products. The development activities used by these disciplines involve physical tools and produce measurable physical changes in the product. The evolving electrical or mechanical product is physical and can be measured at various stages of development.

Software Engineering creates a product much different from the more mature engineering disciplines. The software process creates a non-physical product using activities and tools whose affect on the product is difficult to measure. Unlike other production processes which begin with a wealth of raw materials and transform the raw materials into a product, the software process begins with little or no raw material. As shown in Figure 2.1, process dominates the early phases of software product development while little product exists.

Product measurement alone is difficult until a significant amount of the software product exists. Once a measurable amount of product has been created it is often too late to make needed changes without costly and time consuming rework. For example, if a critical subsystem is found to have a high complexity value during unit testing, little can be done. Redesign of the subsystem is typically not possible within typical budgets and schedules.

Process data is available throughout the software process but the relationship of process measures to specific product characteristics is difficult to substantiate. Process measurement has not reached this level of sophistication. Measurement of both product and process are needed throughout the software process to achieve an accurate assessment.

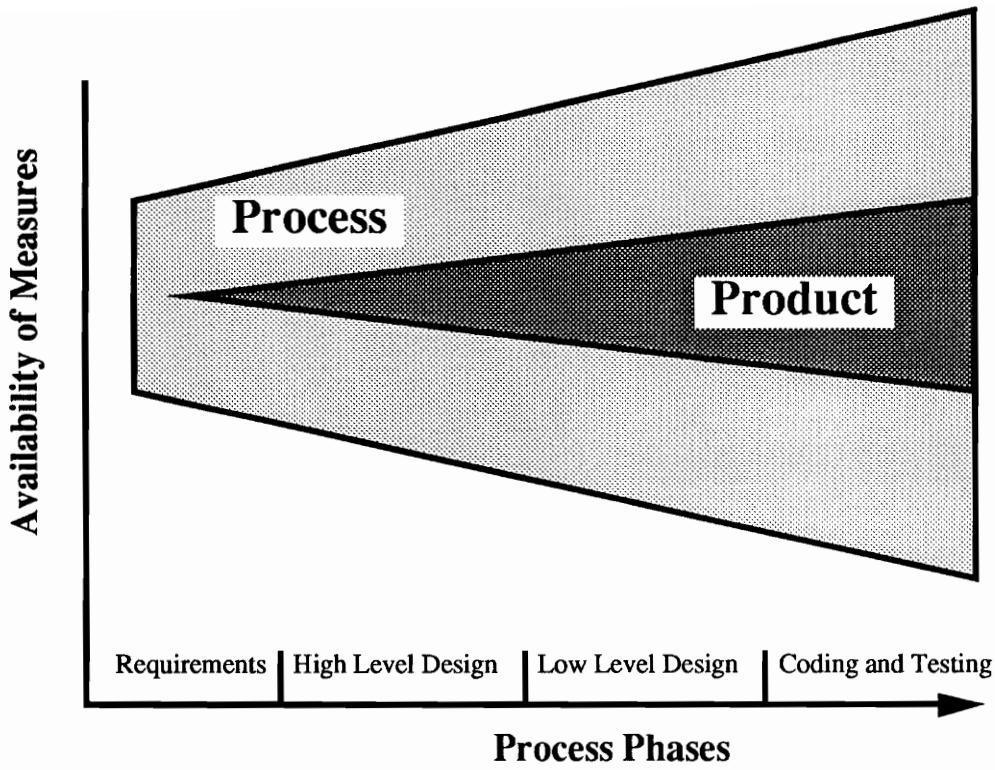

Figure 2.1 Process-Product Relationship Throughout the Process

Many software product measures have been proposed with varying levels of acceptance [COTV88]. Software product complexity measures, based on such properties as information flow, control flow and the number of operator and operands, are the most recognized measures [KAFD82] [MCCT76] [HALM77]. These measures have been used to generate test cases [MCCT90], predict reliability [LEWK88], and predict future product characteristics [HENS90].

The field of software process measurement is not as well developed as software product measurement. The most likely reason for the lack of progress in the area of process measurement is that the software process does not involve activities which are easily amenable to measurement. While, software process activities produce physical changes in the developing product, the effect of these changes is not well understood or easily captured. For example, a single change in the functional requirements may cause a small physical change to the design document but the change may significantly increase the

amount of testing needed. In recognition of this difficulty, Humphrey devotes an entire chapter to "Data Gathering and Analysis," avoiding the terms process measurement and process metrics [HUMW89b]. Process measurement is more correctly in a stage of data gathering and analysis, which will form the basis for selecting and interpreting process measures.

Process data is gathered and analyzed to:

- understand the process,
- evaluate the process,
- control the process,
- predict process or product characteristics,
- evaluate the impact of improvements.

Significant work in process measurement has often focused on software project management data. Schultz proposes Software Management Metrics including process measures in the areas of design, development, testing, and schedule progress [SCHH88]. A software volatility metric is also described, measuring requirements changes over time. These few process metrics can be very valuable, if they are calculated accurately and frequently. In practice determining what work is complete is difficult and highly sensitive to reporting frequency.

The Software Engineering Laboratory (SEL) has expended significant effort in the area of process measurement [SEL90]. The measures are intended to increase understanding of the process and evaluate the effect of methodologies and tools on the software process. The measures described by the SEL capture management information such as cost estimating, scheduling, staffing, effort and progress. Software process measures focusing on software engineering issues include error rates and the rate of software change.

Software measures based on the SEI maturity framework have been proposed by Mogilensky [MOG90]. Mogilensky describes sets of measures applicable to maturity levels 2, 3, 4 and 5. The majority of measures in these sets are product measures evaluated at the conclusion of a phase or activity. The smaller number of process measures evaluate

the effectiveness of an activity at the conclusion of the activity. Interpretation and analysis of the process measures are not discussed [MOG90].

Weiss analyzes software development methodologies through examination of error data [WEID79]. Error rates are dependent on the effort expended and methods used in detection. While errors are an important measure, they are only one characteristic of the software process. Similarly, changes made to baselined code are only a single measure of process effectiveness.

Weiss and Basili investigate the software process by classifying and quantifying change data [WEID85]. Change data provides important insight into the sources of volatility but need to be statistically related to other process data and product characteristics to be valuable predictors of change impact. Tracing changes through the process is also needed.

Arthur et. al. have proposed applying the indicator concept to assess the software process [ARTJ87]. The indicator concept stresses the use of multiple measures to build a body of evidence for the presence or absence of a particular process or product characteristic. While this work shows promise, the effect of process activities on specific software product properties such as coupling and cohesion is difficult to substantiate.

The need exists to collect simple, objective, credible process data capturing both software project management and software engineering information. Rather than proposing process measures which claim to assess, predict, or indicate process characteristics, the more empirical approach of collecting and statistically analyzing process data should be performed. Post-delivery data (data regarding errors detected, effort to maintain, etc.) should be collected as well. Process and post-delivery data can then be statistically analyzed to discover relationships among process data, and between process data and post-delivery product data. Then regression and other statistical techniques are used to assess and predict process and product characteristics in subsequent projects. Process data and post-delivery data from subsequent projects are used as feedback to adjust the statistical relationships.

Analysis Techniques

Many different kinds of process data have been collected to date, including responses to questionnaires, defect data, productivity data and cost data. Both objective and subjective analysis of this data has been performed. Analysis has been performed to compare a process to a particular model [HUMW87], to evaluate product quality [CARD89], or simply to gain insight into the development process [THAR82].

Many researchers have concentrated on change and defect data to analyze the software process [TZEY88] [SHEV85] [WEID85] [WEID79] [CHML90]. Others focus on project management data [THAR82] [SEL90] [HUFK86] [ABDT89] [LANR90]. Both types of data are important for software process assessment, but individually they provide only a single view of the software process. The analysis techniques used by the SEI and those advocated by TQM, based on statistical quality management, are currently receiving the most attention [SEIS92].

The SEI analysis procedure is based on objective criteria, namely the responses to 85 of the 101 questions contained in the assessment questionnaire. An organization is given credit for achieving a process maturity level if 80% of the questions pertaining to that level are answered positively <u>and</u>, 90% of a critical subgroup are also answered positively [HUMW87]. SEI analysis of assessment data can be considered a multi-hurdle analysis, as shown in Figure 2.2.

While the goals of process maturity levels 2 through 5 are desirable, the grading scheme does not completely evaluate the effectiveness of an organization's software process [BOLT91]. For example, an organization could answer 16 of the level 4 questions positively, all 32 of the level 3 questions positively, and all but two of the critical level 2 questions positively, of which 11 are required, and still be rated as level one. Such analysis is clearly flawed.

Levendel advocates the application of statistical process control techniques to the software process [LEVY91]. This approach is based on evaluating the software product quality following each phase of the software process. Quality is based on the results of requirements, design, and code inspection, and unit, integration and acceptance testing.

Evaluating product quality throughout the software process is important but the reliance on inspection and testing as the only criteria is suspect for two reasons. First, defects detected and first-test pass rates are heavily dependent on the thoroughness of inspection [FAGM86] and on test coverage [MYEG79]. Secondly, defects are only one characteristic of product quality [ARTJ86]. Lavendel does not validate any relationships between in-process quality measures and post-delivery characteristics.

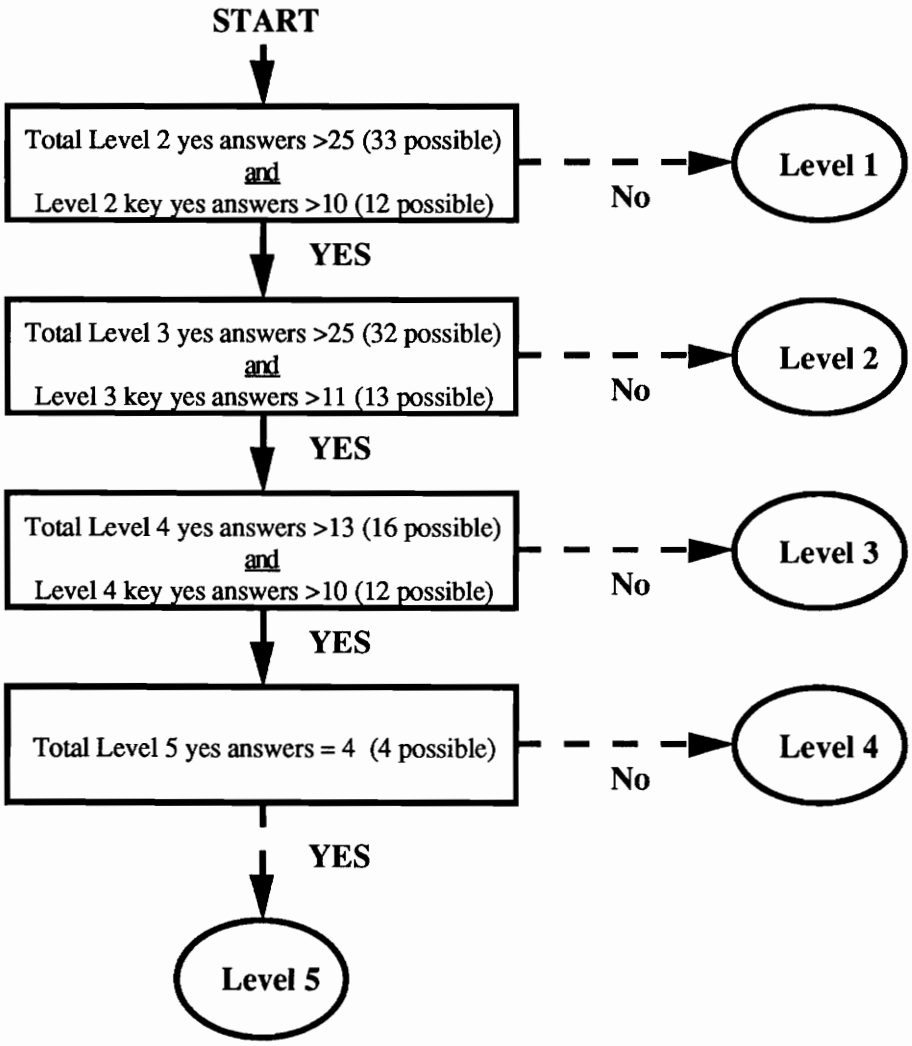

Figure 2.2 SEI Assessment Analysis Technique

Card and Berg describe an industrial engineering approach to software development based on data gathering and statistical analysis [CARD89]. This approach utilizes statistical

process control (SPC) charts in quality control analysis. The concept of applying techniques from SPC is potentially very valuable, although Card and Berg apply it to successive releases of the same product. Application of SPC throughout the software process is needed.

The work of the SEI focuses on the activities present in the software process in relation to their Process Maturity Framework. This type of analysis is critical as the content and structure of process activities is not clear [BASV87]. SEI analysis needs to be supplemented by the use of quality management techniques based on quantitative data, from both management and technical perspectives. Integration of both the SEI process specification assessment and TQM statistical assessment techniques results in a powerful and thorough process assessment approach.

Summary

This chapter described previous work done in the areas of process investigation, modeling, measurement, and analysis. While much work has been done in these areas, a complete, coherent assessment methodology integrating work from each area has not been advanced. Chapter III presents the integrated assessment methodology proposed and implemented during this research effort.

Chapter III. Integrated Assessment Methodology

Introduction

The integrated software process assessment methodology described in this research consists of four steps:

1. Investigate the process.
2. Model the process.
3. Gather data.
4. Analyze the data.

Each of the four steps in the methodology is based on existing methods or techniques. These methods and techniques have been altered and supplemented to increase their effectiveness and to integrate them into a coherent overall approach to process assessment. The results of each step form an integrated, coherent approach to process assessment.

The methodology utilizes both the SEI approach and the TQM approach to process assessment. Investigation and modeling discover and document the activities used by an organization. Data gathering and statistical analysis validate relationships and influences throughout the software process. Consideration of the activities within the process and statistical relationships between and among process and product data determine what types of improvements are needed and how to implement the improvements.

This chapter presents the four steps of the integrated assessment methodology. The first section describes the investigation step. Section two presents the modeling technique used in the second step of the assessment methodology . The third section describes the techniques used to define and categorize assessment data. Section four outlines the techniques used to analyze process content and assessment data. Each step is discussed in detail, and the relationship of the step to the overall assessment is described. Section five summarizes the chapter.

The Investigation Method

An effective process investigation method captures both objective and subjective data. A well-structured investigation obtains data from organizational, functional and behavioral views. Integration of SEI techniques, process audits and modeling views results in a powerful, multi-step investigation method [HENJ91a]. The process investigation method included in this assessment methodology integrates these three areas of process research.

The process investigation method uses the SEI questionnaire, obtaining data which can be analyzed using statistical techniques. Results of the questionnaire are used in a much different way than typically used by an SEI assessment team. Rather than analyzing questionnaire results to determine what answers must be substantiated, the results are used to develop more specific questions used in the follow-up interview. Follow-up questions are created by consideration of process maturity framework goals, software engineering principles and accepted software management practices. These questions investigate the process in a top-down fashion, discovering the activities used in the software process, regardless of their inclusion in the SEI process maturity framework.

The three primary objectives of the investigation method are:

1. To determine the software process used by an organization.
2. To discover the significant factors affecting an organization's software process.
3. To define the methods and techniques used to implement the activities comprising the process.

There are four specific phases to the process investigation method. The first phase in the investigation method determines the staff members to be interviewed. This is done using an initial background questionnaire establishing whether each staff member is directly involved in the software development process, only tangentially involved, or not involved. The SEI assessment questionnaire is used in the second phase to establish the existence of specific activities within the process. The third phase requires follow-up interviews be conducted using more specific questions. The purpose of the interviews is to discover activities not included in the SEI Process Maturity Framework. In addition, follow-up interviews establish how and by whom process activities are performed. The fourth phase

analyzes the follow-up data to determine the differing views of the process within the organization, environmental factors affecting the process and other process activity information. The analysis phase carefully considers organizational factors as well as investigation information to form a correct, complete view of the software process used by an organization.

This investigation method has been successfully performed with excellent results [HENJ91a]. Information about an organization's software process is captured and important problem areas discovered. Many of the recommended improvements have been implemented [HENJ91b] [BOYJ91].

The Modeling Technique

Once a software process is understood, it must be documented to insure consistent use and effective process improvement. Documenting a software process typically results in clarification of the process and the roles of both individuals and groups within the organization. A documented process also aids in establishing what type of data to collect, where to collect the data and what the collected data represent. Process documentation, in the form of a model, is critical to process assessment and improvement.

The software process modeling technique described here graphically documents an organization's software process. The process models produced using this technique must be:

- auditable by both internal and external groups,
- descriptive of the process in multiple levels of detail,
- accurate descriptions of activities within the process,
- maintainable over the long-term,
- defined by significant process phases, and
- understandable by a wide variety of personnel.

The modeling technique is based on control flow diagrams. Control flow diagrams are well suited to modeling software processes because they clearly depict the interactions among process activities. Control flow models of a process are constructed according to specific rules based on the concepts of top-down functional decomposition, information hiding and stepwise refinement.

The modeling technique is based on four major steps, each intended to impart some of the desired characteristics to the resulting models. The four steps are:

1. Three tiers, each with increasing amounts of detail, are used to construct process models.
2. The specific purpose of each tier is defined.
3. Model traceability constraints are specified to insure consistent representation of a process between different tiers.

4. Rules governing the generation of process models through all tiers, and specific rules for the generation of each tier, are defined.

The three tiers used in this modeling technique are shown in Figure 3.1 and described as follows:

1. Tier 1 - Phase Tier. The purpose of Tier 1 is to name each phase within the process. For each phase, the input products and output products, interphase communication, measurements, and the group responsible for implementing the phase are specified.

2. Tier 2 - Task Tier. The purpose of this tier is to describe <u>what</u> major tasks need to be accomplished to implement a phase. A separate task tier exists for each phase.

3. Tier 3 - Procedure Tier. The purpose of this tier is to specify <u>how</u> the major tasks specified in Tier 2 are performed. The major implementation steps for each task shown in Tier 2 are described in Tier 3. A separate procedure tier exists for each phase.

The purpose of each tier is to progressively elaborate the activities within each phase. The tiers relate to each other in specific and well documented ways. Sets of tiers, such as tier 1, tiers 1 and 2, or tiers 1, 2 and 3, form a self-contained graphical representations of process phases. For example, Tiers 1 and 2 shown in Figure 3.1 are used by upper level management to view the process at the task level. These two tiers provide a complete view of the inputs, outputs, measurements, feedback in and feedback out, as well as the tasks involved in the Integration Test and Evaluation Phase. Detailed information about how each task is performed is not needed by upper level management and is not included. These two tiers provide a coherent view of the Phase at the appropriate level of detail.

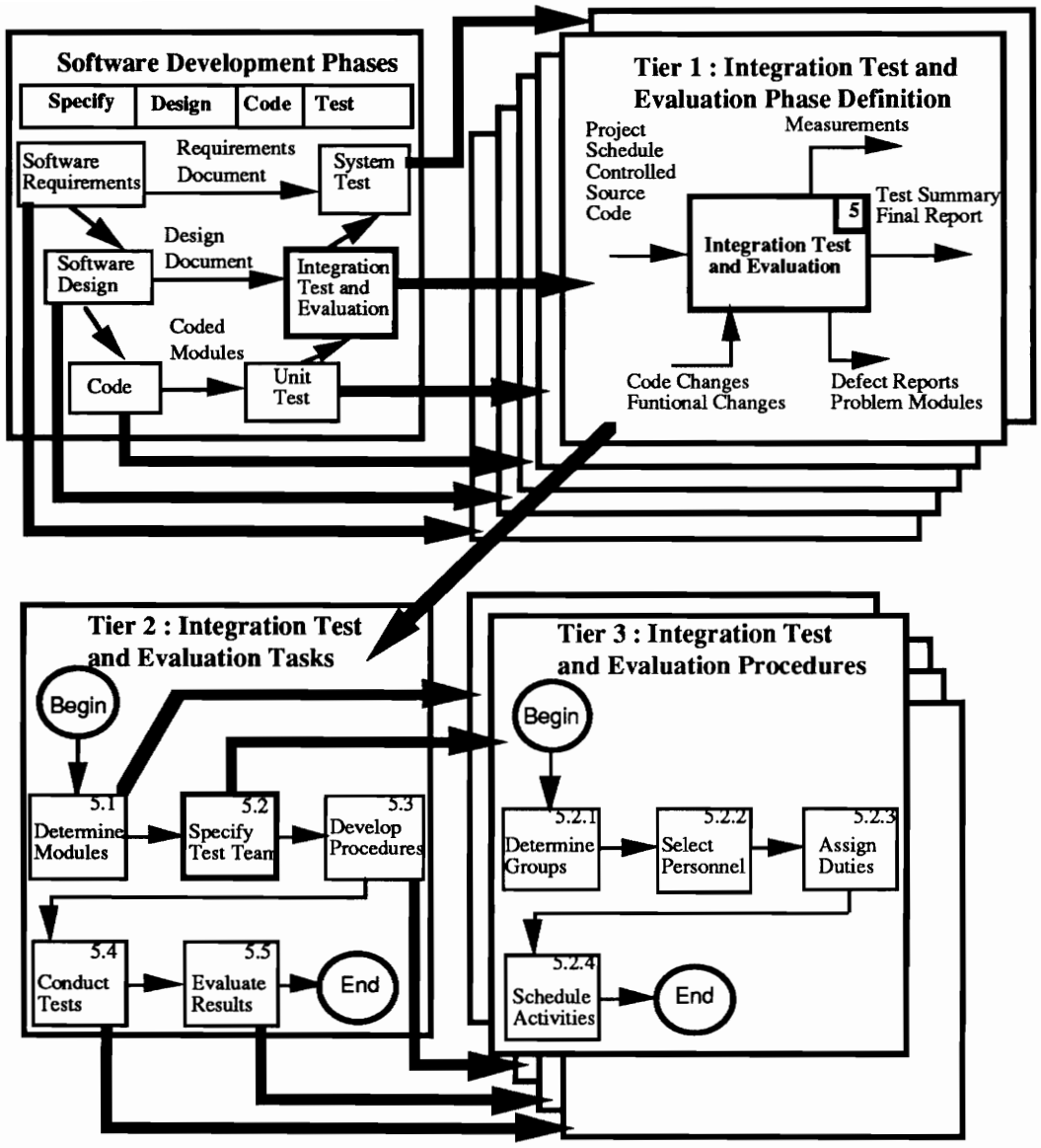

Figure 3.1 Process Modeling Tiers

The traceability constraints insure that each construct shown in every tier has a well-documented, consistent relationship with associated constructs on the next higher tier of abstraction as well as the next lower tier of detail. A structured numbering scheme permits a task to be traced to the procedures implementing the task. Similarly, procedure steps can be unambiguously accumulated into the procedure the steps implement. Verification of the

process representation from tier to tier is supported through additional traceability constraints and generation rules.

A complete listing of the model generation rules is contained in Appendix A. The purpose of the model generation rules is to provide the resulting models with cohesive tiers. The rules are governed by the principles of information hiding, functional decomposition, and stepwise refinement. In addition, the rules encourage progressive elaboration through the more detailed tiers.

Process models are part of a larger effort to define the activities used by the organization. The models are supplemented by additional documentation defining the procedures used by and roles of software quality assurance, independent verification and validation, configuration management and subcontractors.

The modeling technique successfully integrates organizational, functional and behavioral information into a multi-layered process model. The multiple layers provide visibility into the software process used by an organization, as advocated by the SEI. The addition of measurements to the model depicts what type of data is extracted and where. The data is used in statistical process management as required by TQM.

The Data Gathering Techniques

The data gathering techniques described in this dissertation specify how a software process is instrumented. The data gathering techniques:

- classify the data such that it is amenable to statistical analysis,
- define the data clearly and accurately, and
- require review of data definitions.

What data to collect and where in the process to collect the data is dependent on investigation results, the process model and organizational priorities. Process investigation may discover high-priority problem areas within the process which need immediate improvement. The construction of a process model may suggest easily obtainable data about a specific problem within the process. An organization may already recognize a significant, recurring problem within the process needing improvement.

The purpose of this step in the assessment methodology is not to propose a new set of process and product measures but rather to advocate that data gathering be performed. The data is selected by using investigation results, the process model, organizational priorities and previous software measurement work. The data is specified using a defined procedure and a data definition form.

Selection of data to collect should not be performed without a specific purpose and guidelines. The data to be collected must be carefully defined. Haphazard collection of a wealth of data is unlikely to be useful. The assessment methodology requires data to be collected from two different approaches using a data definition form.

Two specific approaches to data collection will be used: horizontal and vertical collection. These approaches are shown in Figure 3.2 and defined as follows:

1. Horizontal - Data about a single activity or product is collected over time or successive process phases to determine characteristics or trends. For example, the number of errors found during successive test phases of a single specification change would be collected.

2. Vertical - Data about a group of activities or products is collected at a single point in time. For example, the estimated number of man-months needed to implement all specification changes approved during Month 2 in Figure 3.2 would be collected.

These two approaches organize the data acquired in several important ways. Vertical data can be viewed as a snapshot of project data at a particular point in time, making the data suitable for use as status and project review data. Horizontal data can be viewed as process and project tracking data, which is useful for discovering trends in the project or process. Classifying data as vertical and horizontal allows the application of statistical analysis techniques to the data. These techniques include analysis of variance, multiple regression, analysis of covariance and categorical independence techniques.

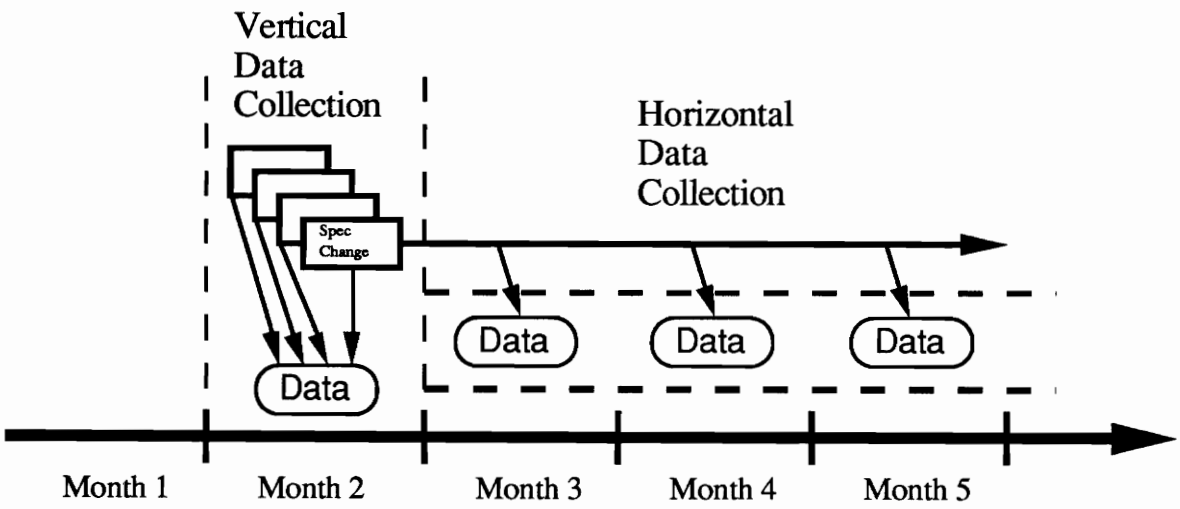

Figure 3.2 Vertical and Horizontal Data Collection

The proposed assessment methodology advocates collection of data based on the work of Boehm [BOEB81], Mogilensky [MOGJ90], SEI [SEI91], Weiss [WEID85] and Huff [HUFK86]. Post-delivery data based on the work of the Rome Air Development Center [CHRA87a], [CHRA87b], Schneidewind [SCHN87], and Rosson [ROSC87], are also suggested. Prospective data should be tailored to the organization and the process used.

Both data definition and tailoring are more easily performed when data descriptions use a defined procedure.

The procedure advocated in this research is as follows:

1. Describe what characteristic the data measures.
2. Define why the characteristic is of interest.
3. Describe how the data will be obtained.
4. Specify what the form the data will take (units, frequency, etc.), and
5. Specify how the data will be validated.

The form used to define the data collected is shown in Figure 3.3.

Characteristic:	This section describes the characteristic of interest.
Justification:	This section describes exactly why the data is collected.
Measurement Approach:	This section describes how and when the data is collected.
Data:	This section specifies the form (units, frequency, formula) of the data.
Validation:	This section describes how the data item is validated.

Figure 3.3 Data Definition Form

An organization's confidence in the validity of the data, and support in the collection of the data, is crucial to obtaining meaningful data. The data definition form specifies the data to be collected in a logical, understandable form. Data definition forms are reviewed by the personnel involved in the software process. Data definition review involves the organization and is critical to gaining the support of software process personnel.

In order to move Software Engineering toward recognition as a true engineering field formal methods must be applied. Scientific investigation utilizes a structured approach to gathering data about an object or process. A structured approach insures the data gathered accurately reflects the characteristics of interest. The vertical and horizontal data-collection approaches structure data gathering. The data definition form and supporting review

insures that the data is clearly specified and completely reviewed. Data gathered using the techniques described here is amenable to many types of statistical analysis methods.

The Analysis Techniques

The goals of the analysis step are to consider both the necessary and sufficient conditions in the assessment. Consideration of the necessary conditions is based on process investigation and the process model. Consideration of the sufficient conditions will be based on the process and product data. Analysis techniques for these two different types of considerations are presented separately.

Process Content Analysis

The information obtained during investigation includes the activities comprising the process and the environmental factors affecting the process. The process model documents the activities used and the structure of the activities within the process. This information answers the following significant questions:

- What activities are missing from an organization's process?
- What activities should be added to the organization's process?
- Where in the process should these activities be added?

Content and structure of the process will be compared with the detailed and extensive Capability Maturity Model for Software (CMM) produced by the SEI [SEI91]. The CMM describes the activities appropriate for process maturity levels 2 though 5. The activities are classified into major categories such as Project Management, Process Management, Software Quality Assurance, Software Configuration Management, etc. While the content and interpretation of the CMM continues to evolve, it does represent a significant repository of accepted software engineering principles and project management practices. The purpose of this comparison is not to force an organization to adhere strictly to the process defined in the CMM, but rather to discover missing activities beneficial to the organization. Previous experience in this area has shown the major benefit of this type of analysis is in the utilization of accepted principles and practices by an organization [HENJ91a] [HENJ91b].

Environmental factors are a significant consideration in the assessment of an organization. These factors often explain why process activities exist, don't exist, or why

a process works in a particular fashion. Previous assessment of small organizations discovered that very different organizational goals significantly impact the software process used [HENJ91a]. For example, the emphasis on error prevention is very different if an organization produces proof of concept software rather than performing long-term maintenance on an established product. Careful consideration of environmental factors allows the appropriate software engineering principles and practices to be selected for inclusion in a software process.

Statistical Analysis of Process Output

Software project managers and division managers typically lack quantitative data regarding project status, progress and quality. Given this lack of objective data, the effect of man-power loss, specification changes, error-rates and other process factors are estimated using rules of thumb, past experiences or "gut feel." Such estimates leave a large margin for error. More mature engineering disciplines rely on quantitative data for determining project status, progress and quality. Software Engineering is in need of quantitative project management based on objective data and statistically valid relationships between and among the data.

Analysis of process output employs accepted statistical methods, such as multiple regression, analysis of variance and covariance and analysis of categorical data. Data acquired during the software process and following delivery of the product are subjected to statistical analysis. Some of the questions which will be empirically evaluated are:

- Can future process and product characteristics be predicted from process data?
- What relationship exists between process data and post-delivery product data?
- What is the relationship between projected data and actual data?
- What is the cumulative affect of horizontal data on the post-delivery product?
- What is the combined affect of process activities on the post-delivery product?

Additional investigation of more specific questions supporting these general questions is also be performed.

The analysis techniques applied in this research project use parametric and nonparametric statistics. Parametric statistical techniques assume data is drawn from a population adhering to a specific type of distribution (e.g., normal). Nonparametric techniques do not assume any specific distribution and are valid across a broad variety of populations [IMAR83]. Nonparametric techniques are advantageous because

- the assumptions are less demanding,
- the affect of noisy data is reduced,
- nominal (assigned to a specific class, i.e., high vs. low) and ordinal (objects are ordered by a greater than or less than relationship) scales may be used, and
- the results are as powerful and rigorous as the results of parametric techniques [SCHN92].

One goal of analysis is to compare the means of samples drawn from two or more independent populations to detect population differences. The Wilcoxen-Mann-Whitney rank sum test and the Kruskal-Wallis test compare means of samples taken from two or more populations [OTTL88]. The test hypothesis in sample mean comparisons is that the sample means are equal. Both tests produce a P-value evaluating the probability of assuming no difference exists between sample means. In this case a small P-value indicates there is ample evidence to support rejection of the test hypothesis. That is, a small P-value indicates a small probability that the populations have identical means.

The association between data values is also of interest. Association is analyzed using linear and rank correlation. Linear correlation measures the linear relationship between two variables. Rank correlation does not assume a linear relationship but rather tests the relationship based on ranks assigned to data values. The rank of a data value is the position of the data value in an ordered list of all data values. In the case of both linear and rank correlation, correlation values range from -1.0 to +1.0. Correlation approaching -1.0 and +1.0 demonstrate a strong association between variables.

A simple classification of software products based on data available early in the maintenance process which predicts future product characteristics is also valuable. For example, classifying software components into categories of error-proneness based on their size is an important classification scheme if a statistically significant association exists

between size and number of defects per component. Contingency tables are used to measure the statistical significance of such associations.

In a contingency table the rows classify items (components) using one criteria (size) and the columns classify items using a second criteria (number of defects). Contingency tables can be used to test the hypothesis that row classification is independent of column classification. This hypothesis is tested using a χ^2 test. A χ^2 value is derived from the classification of actual data. The χ^2 value determines a P-value indicating the likelihood that row and column classification are independent. A small P-value causes the test hypothesis to be rejected in favor of assuming row classification and column classification are dependent. In our example, a small P-value would indicates an association exists between component size and the number of defects per component.

Another goal of analysis is to predict future product and process characteristics using data available at critical points in the baseline effort. The parametric analysis methods of linear and nonlinear regression provide the ability to predict one variable from one or more other variables. The predicted variable (**y**) is referred to as the dependent variable and the variables used to form the prediction equation are called independent variables (**x**). Simple linear regression assumes the dependent variable increases or decreases linearly with respect to one independent variable. Simple linear regression is described by an equation of the form:

$$y = a\,x + b$$

In this equation **y** is the predicted variable of interest, **x** is the variable used to predict **y**, a is the slope of the straight line used to predict **y** from **x**, and b is the y-intercept of the line. The constants a and and b are determined by regression analysis.

Multiple linear regression assumes the dependent variable behaves in a linear fashion, but is influenced by more than one independent variable. Multiple linear regression is described by an equation of the form:

$$y = a_1x_1 + a_2x_2 + a_3x_3 + \ldots b$$

The ability to predict the dependent variable from the independent variables is evaluated in two ways. Associated with each regression analysis is an R^2 value and a P-value. R^2 is the square of the linear correlation coefficient between the actual values and the predicted values. R^2 values range from 0 to 1.0 and represent the amount of variability in the dependent variable that is explained by the independent variables. A large R^2 provides confidence that the prediction equation explains a significant portion of the variability in the dependent variable. The P-value of a test is defined as:

- the smallest probability leading to rejection of the test hypothesis

In the case of linear regression, the test hypothesis is that prediction of the dependent variable using the independent variable is possible. The P-value quantifies the probability of asserting that linear prediction is possible when in fact linear prediction is not possible. Very small P-values represent a low probability of mistakenly assuming prediction is possible. Accurate and confident prediction equations are characterized by both a large R^2 value and a small P-value.

The statistical techniques described here are basic nonparametric and parametric tests available on most commercially available statistical software packages. The results of these statistical tests are easily interpreted and presented. Experience has shown that simple statistical techniques applied to basic data are much more useful in the commercial world than elegant techniques applied to complex metrics.

In an effort to control the overall error-rate, conservative testing techniques were used. The statistical methods previously described also include techniques for controlling experiment-wise error rate.

Summary

The four steps of the integrated process assessment methodology were presented in this chapter. The four steps presented are

1. Investigate the process.
2. Model the process.
3. Gather data.
4. Analyze the data.

Each step was defined and the relationship of each step to the overall assessment methodology was discussed.

The integrated methodology was implemented in a large commercial software organization over a two year period. The implementation effort is described in Chapter IV. Assessment results are presented in Chapter V.

Chapter IV. Implementation of the Assessment Methodology

Introduction

Four research objectives are presented and six significant questions are proposed in Chapter I. Answers to these six questions achieve the research objectives. Chapter III describes the assessment methodology used to answer these questions. Implementation of the assessment methodology must be performed to prove the methodology produces important results and answers the six questions posed.

In order to test the integrated assessment methodology, all four steps of the methodology must be performed for a commercial software organization. The assessment methodology was implemented in a large commercial software organization over a two year period. This Chapter describes the organization, the software product, the assessment focus, and the assessment time period.

The Organization

Implementation of the new integrated assessment methodology described in Chapter III involves four significant steps: investigation, modeling, instrumentation, and analysis. Recognizing the importance of each step, and the need for the assessment team to be present on-site throughout the assessment period, the assessment team specified the following requirements for an participating organization:

1. The organization must be committed to process assessment and improvement.
2. The organization must allow the assessment team on-site access throughout the assessment period.
3. The organization must support the collection of data throughout the assessment period.

The Computer Program Projects group within the Government Electronic Systems Division (GESD) of General Electric (GE) met these requirements. In an effort to improve both the software process and the software products produced, GESD formed the Product Improvement Committee in 1990. The Committee includes four full-time members and over fifty part-time members from throughout GESD. The Computer Program Projects Group agreed to fund an assessment team member for six months per year, over a two year period, for on-site work. In addition, the Group committed to gathering data as a normal part of the software process.

The Software Product

The AEGIS Weapons System is controlled by more than ten different computer programs executing on separate hardware systems. The programs accept input from external sensors and operators. Extensive communication between the programs via a high speed network takes place during operation.

Computer programs controlling AEGIS are written in CMS-2, a relatively unknown and very restricted use computer program. While CMS-2 has been used in other Department of Defense (DOD) projects, it's use outside the DOD world is virtually non-existent. CMS-2 is a block structured language with the ability to manipulate binary data in much the same fashion as assembly languages. For this reason, CMS-2 is well suited to real-time embedded systems and of little use in other situations [BLAR92].

More than ten different computer programs, called elements comprise the AEGIS Weapons System. Each element is responsible for a specific function, such as command and decision, display, communication, data storage, and other important functions.

Element computer programs are decomposed into modules. Modules are collections of procedures. Procedures consist of source lines of code and comments.

Groups of modules which provide a specific functionality are referred to as functional groups. Each element contains several functional groups, each containing a unique set of modules. Modules may belong to only one functional group.

Assessment Focus

The assessment methodology described in this dissertation involves four significant steps, each of which could potentially become a large task in itself. The manpower, resources and time required to assess the entire AEGIS software process are overwhelming for a single researcher. In fact, process assessment and improvement of AEGIS process and product currently involves a full-time committee (the Product Improvement Committee) and a significant budget [GESD91a]. Recognizing the overwhelming task of assessing the entire AEGIS Software Process, and the efforts of the Product Improvement Committee, implementation of the assessment methodology focuses on the perfective maintenance subprocess within the AEGIS Software Process.

Perfective maintenance is defined as:

- enhancements to software which provide additional functionality or modify existing functionality

The tasks employed during perfective maintenance are very similar to those applied during development: specify, design, code, and test. Thus, the first step in perfective maintenance is to obtain written specification of the functional enhancements. The written specification is given by changes and additions to the documentation specifying the functionality of the existing software. In principle the written specification is given completely and is never changed during the ensuing maintenance effort. In practice, however, these specifications are corrected and refined throughout the maintenance process. The changing of functional specifications during maintenance and development is referred to as requirements volatility. Requirements volatility has been cited as the leading problem in a field study of software managers [THAR82]. Changing requirements adversely affects the design, coding and testing of software. An acute need exists to quantitatively assess the perfective maintenance subprocess and the impact of requirements volatility on the perfective maintenance subprocess and the software product.

Perfective maintenance is the focus of this research for three reasons. First, the manpower, resources and time required to assess the entire GE software process were beyond those available in this study. Second, requirements changes, omissions and

misunderstandings are recognized as a major problem in the commercial software industry [BOEB91] [CHML90] [THAR82]. Third, GE recognizes the large amount of effort and resources expended on maintenance. A project level savings of only five percent in maintenance costs would translate into hundreds of thousands of dollars.

Concentrating on a single aspect of the process, such as perfective maintenance, requires the first two steps of the assessment methodology to be implemented for the entire AEGIS Software Process. Results of the investigation step are described in Chapter V and by the author in [HENJ91b]. The modeling step, documenting the GE software process, is overviewed in Chapter V and presented in [HENJ92a]. Full disclosure of the GE software process is not possible here due to proprietary concerns. Discussion of the modeling technique and lessons learned during the modeling process are discussed in [HENJ92c]. The third step in the integrated assessment methodology, instrumentation, took place throughout the assessment effort and are described in [GESD91a], also a proprietary document. Chapter VI presents the results of the statistical analysis step of the integrated assessment methodology.

Assessment Time Period

Implementation of the assessment methodology took nearly two years. Figure 4.1 shows the time-line for implementation of the assessment. Investigation began in May 1991 and concluded in September 1991. Development of the modeling technique took place from May 1991 through July 1991 and was used to produce the AEGIS Computer Program Process Model from September 1991 to December 1991. The Computer Program Projects group gathers specification change data as a normal part of the software process. This data was gathered during the 1991 calender year. New data, based on investigation and the process model were collected both manually and automatically during the 1992 calender year. Data analysis began on-site in April 1992 and concluded in December 1992. Documentation of analysis results took place from August 1992 through January 1993 and are presented in Chapter V.

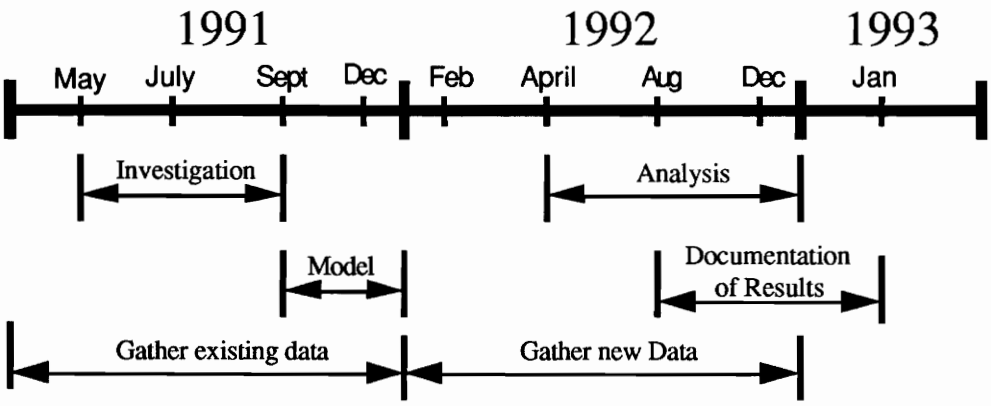

Figure 4.1 Research Time-Line

Summary

This Chapter described the organization assessed using the integrated assessment methodology. The organizational commitment to assessment and improvement, financial support of the author, and assistance in data collection were discussed. The software product was introduced, including the implementation language and structural breakdown. In addition, the focus of this research was motivated and clearly specified. A complete time-line presenting the historical progress of this research was also shown and described.

Chapter V. Process Content Analysis Results

Introduction

This chapter describes the software process used by GE to develop and maintain AEGIS software and the process used to specify, change and implement functional enhancements to existing AEGIS software. The first section describes the GE software process based on results of the investigation step, presents an overview of the process model, and concludes with process content analysis results. The perfective maintenance process, used to specify and implement functional enhancements, is described in the second section. The second section concludes with results of content analysis of the perfective maintenance process. The third section summarizes the results of the content analysis step of the integrated assessment methodology.

GE Software Process

A single software process is used by GE for both maintenance and development. The process follows the basic waterfall model of software development, namely requirements specification, design, code and test. The largest and most significant software product GE maintains is the AEGIS weapons system.

GE manages maintenance of AEGIS software by coordinating the work of many internal groups as well as a software subcontractor. GE System Engineering analyzes all functional upgrades and makes all major technical decisions concerning both hardware and software. The Computer Program Projects Group manages the fiscal, manpower, and equipment resources for AEGIS software projects. In addition, the Computer Program Projects Group manages software subcontractor testing and performs two testing phases. GE subcontracts Computer Sciences Corporation (CSC) to perform detailed design and coding of AEGIS software. Software Quality Assurance, Configuration Management and Computer Program Development groups within both GE and CSC are also involved in AEGIS software maintenance. Raytheon Corporation performs minor software subcontracting as well.

The U.S. Navy is actively involved in requesting upgrades and reviewing product and process throughout maintenance. Completed AEGIS software, including both code and documentation, is delivered to the Navy at the Naval Systems Warfare Center in Dahlgren, Virginia. The Navy distributes AEGIS software to ships throughout the world and maintains the code and documentation following delivery.

GE Software Process Description

AEGIS software is developed and maintained in compliance with Department of Defense (DOD) Military Standard 1679a. This standard requires many products and activities. Most importantly, twelve process phases are required when developing or maintaining software. The twelve phases are presented in Figure 5.1. Complete specification of the GE software process is company proprietary and cannot be presented here.

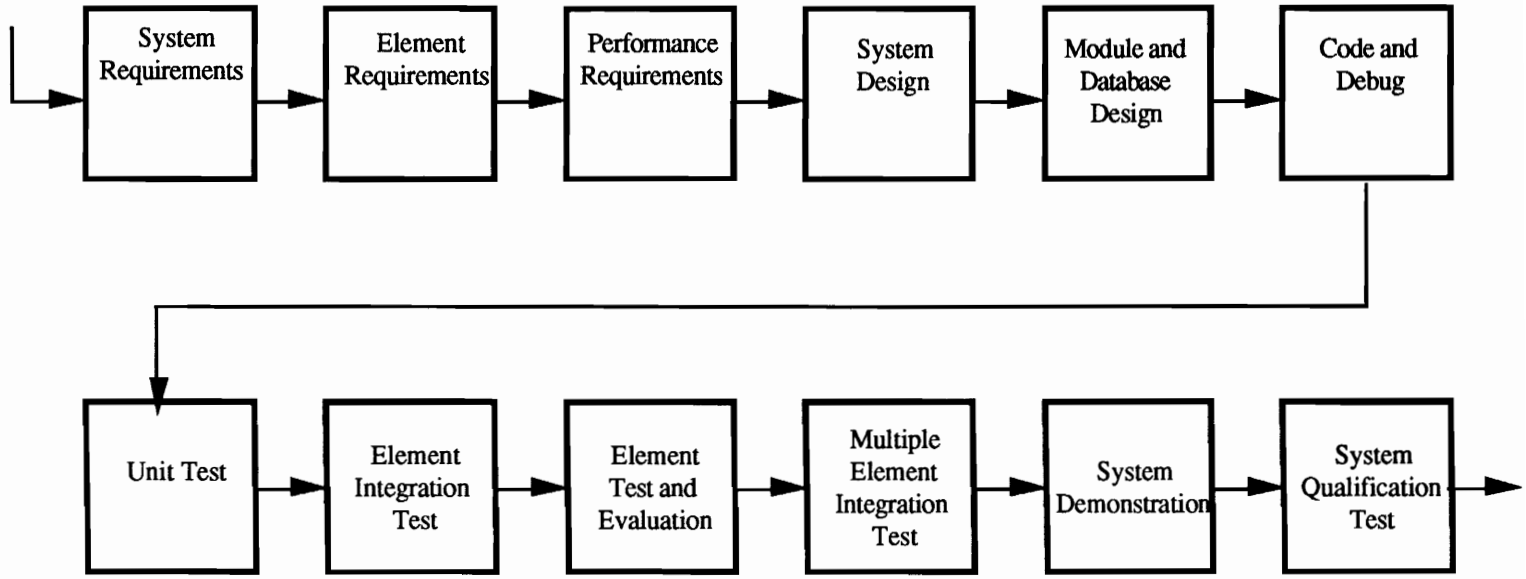

Figure 5.1 GE Software Process

System requirements is the first phase. The purpose of this phase is to define high-level requirements for the entire system. Requirements defined in this phase are allocated to hardware, software, or operator. System requirements are specified in a reviewed and controlled document referred to as the System Specification.

Phase 2, Element Requirements, requires further analysis and specification of system requirements allocated to system software. Software requirements are assigned to software elements comprising the system software. Software elements are typically separate computer programs providing specific functionality to the target system. Examples of software elements are a database manager, radar tracking software, or a command and decision program. Element specification documents describe requirements allocated to each software element.

The Performance Requirements phase involves specification of software performance requirements. Embedded real-time software such as those software elements included in the AEGIS system have important response time and memory usage requirements. These requirements significantly impact the design and construction of software elements. Performance Requirements are specified in the software element Program Performance Specification.

High-level software design takes places in the Design phase. Each software element is designed using functional decomposition as required by DOD Military Standard 1679a. The Design phase assigns software element requirements to groups of modules within each element. These groups are referred to as functional groups and are comprised of software modules. The Design phase concludes with a Critical Design Review.

Module and Database Design is the fifth phase of the GE software process. Requirements assigned to functional groups during the Design phase are further assigned to modules within each functional group in the Module and Database Design phase. Database storage and access requirements are specified in this phase as well. Both GE and CSC personnel participate in the Module and Database Design phase.

CSC software developers create and modify source code in the Code and Debug phase. Specification and design documents drive this phase. GE project managers monitor the

Code and Debug phase through periodic meetings and status reports. GE and CSC upper-level management receive oral monthly status reports in a formal setting. Coded modules placed in a source code library result from this phase.

Unit Test is the seventh phase of the GE software process. CSC performs unit testing on coded modules from the source code library. Modules passing unit testing are placed in a source code library controlled by CSC configuration management. Quality assurance participates in this phase, insuring unit test scripts, source code and supporting documentation meet existing standards.

Elements comprised of unit tested modules are tested in the Element Integration and Test phase. Elements are first created through integration of modules. Integration errors detected are documented and corrected. Integrated elements undergo integration testing under the direction of CSC software developers, configuration management, and software quality assurance.

Elements passing integration test enter the Element Test and Evaluation phase. GE test personnel install the elements on computers at a GE test site. This site simulates the environment of a naval ship. Elements are tested to determine if element software requirements are met. Individual requirements from the element specification documents are tested, results are documented and a final test report is prepared. Defects found in this phase are reported and corrected as time permits.

Software elements are integrated and tested in the Multiple Element Integration and Test phase. Software elements are integrated into the single software system used to control the complete system. The software system requirements drive testing of the software system in the Multiple Element Integration and Test phase.

The Demonstration phase is the first phase with high visibility by the customer, namely the U.S. Navy. GE, CSC and naval personnel observe a demonstration of the system at the GE test site. Reliability and maintainability information is recorded as well as defects during the Demonstration phase.

System Qualification Test takes place at the Naval Surface Warfare Center in Dahlgren Virginia. Testing takes place aboard a U.S. Navy ship assuring the system meets the minimal criteria for installation and deployment. This phase completes the GE software process and concludes with release of both code and documentation to U.S. Navy personnel.

GE Software Process Content Analysis

Software process content analysis is performed by comparing an organization's software process to the SEI process maturity framework and to established software engineering principles. Analysis of the GE software process included an estimated SEI assessment and follow-up investigation as specified in Chapter III. The results of this investigation are presented here. In response to this analysis an improvement plan was written [HENJ91b].

Time, resource and economic constraints prevented a full SEI assessment from being conducted. An estimated SEI assessment was performed through the following activities:

- review of product and process documentation (i.e., standards, directives, product work instructions, etc.)
- interviews of GE and subcontractor personnel

These activities provided estimated answers to the assessment questions and additional information about the GE software process. Follow-up investigation targeted the specification change process described in the following section.

Results of the estimated SEI assessment are as follows:

- SEI level 2 questions - 32 positive answers (76%) and 10 negative answers (24%)
- SEI level 3 questions - 15 positive answers (39%) and 19 negative answers (61%)

The negative answers to questions from the assessment questionnaire were used to identify areas within the GE software process needing improvement and further investigation. Negative answers were also grouped by Key Process Areas (KPAs) within the SEI

Capability Maturity Model. Table 5.1 summarizes the results of the estimated SEI assessment by question category, number of questions with negative answers, and the KPAs associated with the questions. Each question category is discussed in the following sections.

Table 5.1 Estimated Assessment Results

Question Category	Number of Questions with Negative Answers	Key Process Area (Maturity Level)
Training	5	Training Program (Level 3)
Project Management	6	Software Project Management, Tracking and Oversight (Level 2)
Development Processes	8	Organization Process Definition (Level 3)
Testing	4	Software Product Engineering (Level 3)
Requirements Traceability	3	Requirements Management (Level 2)
Subcontractor	3	Software Subcontractor Management (Level 2)

Training

Current training practices within the AEGIS development community include introductory courses for engineers, a functional analysis course and a software project management course. GE personnel completing these courses generally have positive comments regarding course content and utility in their job activities.

While the training courses are seen as beneficial, several problem areas exist. Not all staff involved in engineering, testing and management are required to complete these courses. The existing courses fail to target two areas specifically called for in the SEI process maturity framework, peer review leadership and design paradigms. Many of the existing AEGIS training courses are valuable stand-alone courses but do not integrate into the current GE software process.

Project Management

Project management of AEGIS software is a complex and difficult task because of the number and parallel timing of multiple maintenance efforts. The practices employed to plan and track the progress of multiple efforts have improved in response to the problems encountered.

Despite improvement of AEGIS project management practices, several problem areas exist. No single database of size, schedule and cost data exists for AEGIS baseline development. The lack of a central repository for past and current project data prevents introduction of valuable project management procedures and tools requiring this data. In addition, quantitative tracking of software projects and project to project improvements are difficult to accomplish. The present tracking system also does not record project data at the level of granularity needed to leverage past experience into process improvement on future projects.

Development Processes

A major effort ongoing by the Process Improvement Committee (PIC) is the implementation of the documented GE software process. Computer program development processes have been documented but must be implemented by AEGIS software projects throughout the organization. Process standards have been reviewed and a major metrics effort has been launched. Completion of these efforts by the PIC corrects the problem areas described in this section.

While AEGIS software is developed and maintained under DOD Military Standard 1679, a great deal of variability in the actual process used to produce deliverable code and documentation is possible. In fact, a great deal of difference exists among AEGIS projects according to time, project management and project personnel. The defined software process is not implemented across the organization. Process verification must be performed to insure the process is implemented as documented or that logical exceptions are granted.

There is also a need to establish a Software Engineering Process Group (SEPG). SEI level three criteria specifically calls for a SEPG. An SEPG responsible for process definition, change and implementation is necessary to maintain consistent focus on improvement of the development process. The PIC currently performs many of the functions allocated to a SEPG. PIC efforts have established the basis needed to measure and improve the GE software process. Continued focus on the GE software process is the responsibility of a SEPG.

A single central repository of metrics data does not exist. SEI level three criteria specifically requires a metrics database for storage, manipulation and analysis of metrics data. A metrics database is under development at this time.

A documented plan for analyzing metrics data and distributing results is also lacking. The analysis techniques proposed in this dissertation and the assessment results are to form the basis for the GE metrics analysis plan.

The development and maintenance of documentation is another problem area within the GE software process. GE technical writers create and maintain system and software documentation on a large mainframe without any configuration control system. No software tools exist to capture changes or store successive versions. Follow-up investigation discovered that documentation is often delivered months, even as much as a year after code delivery. Clearly an automated documentation system including configuration management capabilities and support tools is needed.

Testing

SEI maturity level 3 requires documented standards for:

- planning testing phases
- generating test procedures and test cases
- conducting peer reviews of test procedures and test cases

Tool support of testing phases is also required to meet level 3 criteria. Test phase planning and test case generation are areas where tool support is most beneficial.

The AEGIS Engineering Test and Evaluation (ET & E) phase is implemented in accordance with Naval Systems Instruction NS-I-076-A. This instruction provides a basis for the testing phase but does not contain or refer to standards for planning the test phase or generating test procedures test cases. Peer reviews of test procedures and test cases are not performed.

ET & E test engineers employ an accepted set of undocumented test phase planning practices. Test cases and test procedures are also generated using standard practices. Documented standards for test phase planning and test procedures do not exist. These standards insure consistency across AEGIS software projects and provide a basis for test case peer reviews.

Unit testing and Multiple Element Integration and Test also need to be documented to meet SEI maturity level 3. The problem areas described previously also exist in these two subcontractor-performed test phases.

Requirements Traceability

SEI level 3 criteria requires a procedure for maintaining the traceability of requirements from specification documents to design documents, and from design documents to source code. Tools supporting traceability are also included in the level 3 criteria.

Presently limited traceability of software requirements exists. Manual traceability of requirements to test cases is performed. Groups of modules within an element are identified as providing specific functionality. This is a form of requirements tracing. An integrated traceability procedure does not exist to trace requirements to design documents, and from design documents to source code.

Requirements tracing can be traced to source code only through a significant manual effort. Functional upgrades to existing AEGIS software elements are coded with an alphabetic mnemonic and assigned a specific number to be included in the comment present on all lines of code added and changed. Changes to functional upgrades occurring during a

maintenance effort are also assigned a specific number. Each group responsible for an element records and maintains these numbers but no central database exists.

Requirements are not traced to design documents. Follow-up investigation revealed design documents rarely reflect current system design and contain little or no reference to specification documents.

Timely, cost-efficient maintenance of requirements traceability is lacking. Requirements traceability must be performed according to a documented procedure and supported by software tools.

Subcontractor

GE software subcontractors, chiefly CSC, currently follow DOD Military Standard 1679, a product-oriented software process. DOD Military Standard 1679 mandates products be delivered at specific intervals during the software process. The process is product-oriented because of the heavy reliance on documents as a means of proving the correct process is implemented. While GE software subcontractors deliver all documentation required, the process used to develop and maintain software is often much different from that specified in DOD Military Standard 1679.

The defined GE software process includes processes implemented by software subcontractors. SEI level 3 criteria require all subcontractors to follow a documented process which is also a maturity level 3 process. No mechanism exists to verify adherence to the documented process by the subcontractor. In addition, process and product metrics specified by [GESD91a] are not contractually required and therefore not supplied.

A consistent approach to subcontractor performance tracking and review based on the process definitions does not exist. Subcontractor performance tracking and review are currently performed by many groups, including Software Quality Assurance, Computer Program Projects and System Engineering. In-process reviews, monthly project status meetings and product inspections track and review subcontractor performance. Follow-up interviews revealed a great deal of confusion and anxiety among subcontractors regarding the procedures and practices used to track and review projects. In short, subcontractors

perceived themselves as subject to different tracking and review procedures from project to project.

GE Software Process Content Analysis Conclusions

Investigation of the GE software process required significant effort over a six month period. SEI process assessment provided evidence of the presence or absence of the specific process activities contained in the SEI process maturity framework. Important process assessment data resulted from the estimated SEI assessment. Follow-up investigation provided additional significant data regarding procedures and practices not detected by the SEI assessment method.

The author produced an improvement plan specifically intended to improve GE to SEI maturity level 3 [HENJ91b]. While little progress has been made in the area of training, efforts are currently underway in each of the other areas described. In fact, an SEI assessment performed by an SEI licensed organization is scheduled for June 1993, and will involve the author. Such an assessment will likely result in GE achieving SEI maturity level 3 due to the efforts of the PIC.

Perfective Maintenance Process

The assessment conducted in this dissertation concentrated on perfective maintenance (adding new functionality to existing software) activities within the GE software process for three reasons. First, the manpower, resources and time required to assess the entire GE software process were beyond those available in this research effort. Second, requirements changes, omissions and misunderstandings are recognized as a major problem in the commercial software industry [BOEB91] [CHML90] [THAR82]. Third, GE recognizes the large amount of effort and resources expended on maintenance. A project level savings of only five percent would translate into hundreds of thousands of dollars.

The AEGIS weapons system has undergone extensive maintenance during it's twenty-two year lifetime. The maintenance efforts are largely aimed at incorporating new functionality. New functionality must be specified in existing software specification documents and implemented in the element computer programs. The specification change process assessed in detail in this dissertation includes all activities performed to specify, review and implement new functionality in existing AEGIS computer programs.

Perfective Maintenance Process Description

The element computer programs comprising the AEGIS weapons system are released in baselines. A baseline is implemented under a single contract which specifies a set of functional enhancements. Each enhancement is referred to as an upgrade item. Upgrade items typically require changes to the Program Performance Specification (PPS), the Interface Design Specification (IDS) or both. The process used to create new baselines is shown in Figure 5.2.

System engineers analyze the impact and interpret the precise meaning of each upgrade item during the system specification and software specification phases, depicted together as Specification Analysis in Figure 5.2. This analysis is documented in the new PPS and the new IDS, which are reviewed at the Preliminary Design Review (PDR). The new PPS and new IDS are the basis for implementing the new baseline. Hereafter, PPS and IDS refer to the new versions of these documents.

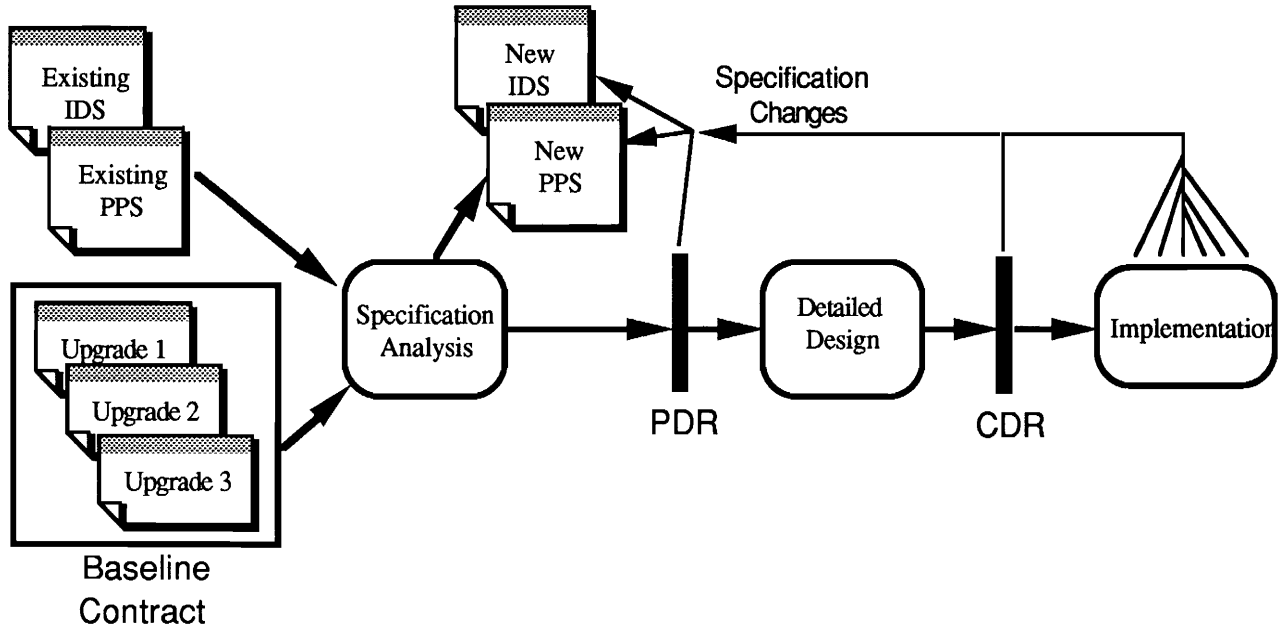

Figure 5.2 Specification Change Process Overview

Detailed design of upgrade items takes place following the PDR. Detailed analysis and design of the functionality documented in the PPS and IDS typically require additions and changes to the PPS and the IDS. The IDS and PPS pages requiring changes and the detailed design are reviewed at the Critical Design Review (CDR). Successful completion of the CDR, as judged by the customer representatives and management, initiates the implementation phase.

The new versions of the PPS and the IDS presented at the PDR and the CDR are modified at the PDR, the CDR, and post-CDR. These specification changes are documented using a specification change form and implemented following the CDR.

Specification changes are the driving force behind each baseline. Specification changes are a major source of difficulty in producing a new baseline because

- the majority of specification changes occur after CDR,
- most specification changes require code changes,
- implementation requires coordination of system engineers, project managers, subcontractors, testers and technical writers, and
- the timing of specification changes requires a functionality to be fit into an already designed product.

These issues make specification changes a serious concern of each baseline effort.

The PPS and the IDS are written, maintained and controlled by the system engineering group. This group determines the need for a specification change and originates a change using a specification change form. The completed specification change form has attached to it copies of the PPS or IDS pages requiring change. The specification change form and attached pages are reviewed and approved by both management and the customer before implementation is begun.

Specification changes submitted at the PDR and the CDR are evaluated and directed during these reviews. A specification change may be directed for implementation

(approved), assigned to a future baseline (deferred) or rejected (withdrawn). Specification changes approved at the PDR and the CDR are implemented following the CDR.

Specification changes originating after the CDR are submitted to a review board. The board consists of representatives of project management, system engineering, configuration management and the customer. A specification change may be approved, withdrawn or deferred to another baseline at the direction of the board. The board is supplied with estimates of cost and schedule impact as well as the estimated impact on computer programs.

Specification changes approved for implementation at the PDR, the CDR and by the review board are assigned a computer program change number. This number is entered as a comment on each line of code added or changed to implement the associated specification change. In addition, the programmers implementing specification changes record the number of lines of code added, deleted and changed. The specification change and computer program change number provide traceability of each specification change. Detailed information about implementation effort, test results, and final status is also recorded for each specification change.

Perfective Maintenance Process Analysis

The perfective maintenance process described in the previous section appears well-specified and tightly controlled. In fact much variability exists in generating, analyzing and implementing specification changes. Eight different GE directives and two military standards, DOD Military Standard 1679 and DOD Military Standard 480 (Configuration Management) discuss or are related to processing specification changes. No single standard or directive exists specifying the generation, analysis and implementation activities for changes to software specification documents. Without a documented specification change process software project managers and personnel have no clear direction on managing and implementing specification changes.

The lack of a documented specification change process causes many problems during a baseline effort. In some instances computer program change numbers are used for multiple specification changes. This significantly reduces the ability to trace the impact of

specification changes on computer programs. In addition, the accuracy of the estimated impact of each specification change on computer programs cannot be judged.

Specification change forms and the attached pages from software specification documents are sometimes revised after a specification change has been approved for implementation. These revisions significantly change the cost and schedule impact of specification changes. Revised specification changes require new paper copies of the specification change form and attached change pages be distributed and old copies discarded. Multiple generations of specification change forms are a major problem throughout the groups and organizations maintaining AEGIS software.

A serious problem occurs when a specification change is approved by the review board but the software subcontractor is not directed to implement the specification change. In this case, the software specification document includes the specification change when the computer programs do not provide the functionality.

Specification changes follow a lengthy paper trail resulting in delays of four to six weeks between specification and the start of implementation in the element computer programs. This paper trail often prevents specification changes from reaching all the personnel needed to process the change.

No single central repository of specification change data exists, making status tracking and reporting difficult and inaccurate. Each group maintains a database of specification changes, including status, implementation effort estimates, and relationships to upgrade items and other specification changes. Multiple databases spread throughout the organization results in inconsistent views of the progress and status of maintenance efforts.

Finally, no single method for identifying specification changes exists. Specification changes are assigned identification numbers by Systems Engineering, the software subcontractor, and Configuration Management. The use of multiple numbering systems results in tracking and reporting problems.

In short, the lack of a documented specification change process at GE at the time this research effort began was a serious problem. The specification change process required

investigation, documentation and measurement in order to understand, assess and improve the process. Consequently, this research effort focused on perfective maintenance and specification changes.

Perfective Maintenance Process Analysis Conclusions

Investigation of the perfective maintenance process and the specification change process revealed many problems which explain the significant amount of difficulty experienced implementing and tracking specification changes. These problems result from the inherent complexity of AEGIS documentation and computer programs, and the GE organization.

Investigation and analysis results presented in the previous section have been delivered to GE and action has been taken to correct the problems discovered. A standard specification change process has been documented by the author, reviewed and approved within GE managment, and is now in use. Appendix B contains a copy of this standard. A distributed database is currently under development for storage of specification change status and tracking information. This database is based on the specification change process documented in Appendix B and captures much of the data described in Chapter VI.

Investigation and analysis of the perfective maintenance process has provided important insight into the GE software process and software process assessment as well. GE has realized substantial benefit from this research. The perfective maintenance process has been investigated and analyzed across GE groups and software subcontractors. The existence of a single documented specification change process provides a standard for each group and organization to follow. In addition, the results described here prove the investigation, modeling, and process content analysis techniques presented in this dissertation are effective in assessing an organization's software process.

Conclusions

This chapter presented the results of process content analysis of both the GE software process and the perfective maintenance process. Investigation results were presented as well the results of process content analysis. Results presented in this chapter show the integrated process assessment methodology is very effective in discovering and analyzing an organization's software process. The second analysis technique, statistical analysis of process output, is presented in Chapter VI.

Chapter VI. Statistical Analysis Results

Introduction

This chapter describes the results of applying statistical analysis techniques to process and product data collected throughout a specific baseline effort. The first section presents statistical results of the impact of the GE perfective maintenance process on the software product. Important product analysis results in three areas are presented: module impact, incomplete computer program changes, and post-delivery product impact. Statistical analysis results of the GE perfective maintenance process are described in the second section. Three important analysis results in the areas of upgrade item effort, specification change productivity, and the effectiveness of the Engineering Test and Evaluation phase are presented.

Product Analysis

A large amount of data about upgrade items and upgrade specification changes impacting the largest computer program in the AEGIS weapons system, the command and decision element, was captured during a specific baseline (Baseline 4 Phase II). Data was collected during the baseline effort by the project management organization, the software subcontractor, the metrics group, testing groups and upper level management. Validation was done by cross referencing data, performing an automated analysis of the delivered source code, and interviewing personnel.

The product data captured during Baseline 4 Phase II is of several different types, namely:

- change of module and element computer programs in source lines of code (SLOCs)
- module and element size
- detected defects per module and per element
- number of upgrade items and upgrade specification changes impacting each module and element

Validated data is stored and analyzed using the Exstatixs statistical analysis software package. This package allows data to be input, manipulated, graphed and statistically analyzed in many different ways. The results produced by the software package were verified using the Minitab statistical package.

Upgrade Impact on Software Modules

All SLOCs added or changed to implement a specification change are commented with a computer program change number. In addition, all lines of source code added or changed to correct errors in the AEGIS computer programs are similarly commented. These computer program change numbers provide a way to trace the effect of specification changes and error corrections to AEGIS computer source code. Changes to the command and decision element were captured using these computer program change numbers.

The command and decision element is comprised of modules which are made up of procedures. Procedures consist of source code and comments. Computer program change numbers entered on each changed or added line of code allow modifications to be traced to, and accumulated by, procedure and module.

The relationship between upgrade items, upgrade specification changes and errors corrected per module in the control program was investigated. Specifically, the impact of upgrade items and upgrade specification changes on the reliability of modules within the control program was analyzed. Reliability in this context means the error-proneness of modules and was measured by errors corrected per module. Finally, if relationships existed, the relationships were to be quantified.

The number of errors corrected per module was captured as well as the number of upgrade items and upgrade specification changes affecting each module. The number of changed lines of code within the module, and the number of persondays expended per module to implement upgrade specification changes were also captured.

Simple graphs were constructed to compare the impact of upgrade items and upgrade specification changes to the reliability of software modules. The graphs indicated nonlinear relationships existed, suggesting non-linear correlation or rank order correlation be applied. Given GE management viewed non-linear correlation with some level of distrust regardless of the results, rank correlation analysis was performed.

Rank correlation measures the association between errors corrected per module and the impact of upgrade items per module. Impact on modules is measured by the number of upgrades affecting a module, the number of upgrade specification changes affecting a module, and the percentage of the module changed to implement upgrade specification changes. Impact on the maintenance process is measured by the number of persondays expended implementing upgrade specification changes per module. Strong rank correlation existed between errors corrected per module and upgrade item impact, as shown by the rank correlation coefficients in Table 6.1.

A simple classification of modules according to upgrade impact which predicts modules as having more or less than the median number of errors was also desired. Contingency

Table 6.1 Correlation Between Errors Corrected and Upgrade Impact

	Number of Upgrades	Number of Upgrade Spec Changes	Percentage of Module Changed	Number of Persondays
Number of Errors Corrected	.803	.855	.767	.808

Table 6.2 Number of Defects vs. Upgrade Item Impact

	<= Median Number of Upgrade Items Impacting	> Median Number of Upgrade Items Impacting
<= Median Number of Errors Corrected	22	5
> Median Number of Errors Corrected	2	26

Table 6.3 Number of Defects vs. Upgrade Specification Change Impact

	<= Median Number of Upgrade Spec Changes Impacting	> Median Number of Upgrade Spec Changes Impacting
<= Median Number of Errors Corrected	25	3
> Median Number of Errors Corrected	6	22

tables, as described in Chapter III, were created using median values of errors corrected, median number of upgrades affecting a module and median number of upgrade specification changes affecting a module. The contingency tables test the null hypothesis that there is a relationship between the number of upgrade items and the number of errors corrected in program modules, and between the number of specification changes and the number of errors corrected in program modules. The contingency tables are shown in Tables 6.2 and 6.3, and both have P-values of less than 0.005, indicating a very strong evidence against rejecting the null hypothesis. Statistically, these contingency tables support the assertion that more than the median number of errors are corrected in program

modules impacted by more than the median number of upgrade items and specification changes.

The rank correlation coefficient suggests that as upgrade item impact increases the number of errors increases. Strong correlation suggests modules can be ranked using upgrade impact. This ranking predicts the ranking of modules based on errors corrected. Modules predicted to be error-prone can be selected for module inspection following the coding phase in an attempt to detect errors prior to lengthy and expensive test phases. Module ranking can also be used to select modules for additional unit and module integration testing prior to system testing.

Table 6.2 shows that selecting modules based on upgrade item impact greater than the median number would include 26 of the 28 modules (93%) with error rates above the median number of errors. Of the 31 modules selected with upgrade impact greater than the median value, 26 of them (84%) actually had more than the median number of errors per module.

The analysis quantitatively evaluates the impact of upgrade items on modules. Modules can be rank ordered based on upgrade item impact or the historical median value of upgrade item impact can be used to classify modules as more or less error-prone. Upgrade impact information is currently being used within the organization to select modules for additional code walkthroughs or more intensive testing.

Incomplete Program Changes

Changes to computer programs are generated in response to errors and specification changes. A unique computer program change number is assigned to implement each change to computer programs and is used to track changes to source computer programs. Implementation information about computer program changes is stored in a database and tracked using specific codes to describe the status of each change. A change may be open (under development), or closed (coded and successfully tested).

Computer program changes remaining open against a computer program at the completion of the product baseline represent unfixed errors or unimplemented specification

changes. These changes and errors are deferred to future baselines. The number of open changes is a concern to both the organization and the customer at the delivery of the product baseline.

The goal of analyzing open changes is to determine if the number of open changes can be predicted using data available during development. The accuracy and timeliness of the predictions must also be determined. If such predictions can be accurately produced, the predictions could be used to limit the number of changes approved for a baseline and thus reduce the number of open changes at the delivery of a baseline.

Applying the technique of multiple regression, prediction equations were fitted using data available throughout the maintenance process. Table 6.4 presents the results of using multiple regression to predict the number of open computer program changes for the 11 computer programs in the weapons system.

Table 6.4 Results of Open Computer Program Change Regression Analysis

Dependent Variable	Independent Variables	R^2	P-Value
Open Computer Program Changes	• Spec Changes Impacting Code • Spec Changes not Impacting Code • Interface Change Requests	.99	.0016

The independent variables used in the equation are generated throughout the maintenance process. The equation generated can be applied with the available data to provide predictions at specific points during the implementation effort. These predictions can be used by project management to quantitatively estimate the number of open computer program changes at the conclusion of the current baseline. The R^2 of .99 and the P-value of .0016 provide a great deal of statistical confidence in the regression equation. The data set is small in this case (11 different computer programs), making further analysis across baselines necessary.

Post-Delivery Product Impact

AEGIS weapons system software maintained by GE is delivered to the U.S. Navy at the Naval Systems Warfare Center (NSWC) in Dahlgren, Virginia. Delivery of an AEGIS baseline terminates the contract between GE and the U.S. Navy. After delivery, AEGIS software is distributed and installed on Navy ships world-wide. Delivered AEGIS software is maintained by Naval and civilian technical staff at Dahlgren.

Defects found during installation and use of AEGIS computer programs are reported to NSWC. Defect data collected and analyzed by NSWC is stored in a large database at Dahlgren. Defect data are collected by functional group within each of the thirteen computer programs comprising the AEGIS weapons system. Defect data includes an identification number, description, computer program, functional group, date reported, and priority (ranging from 1 to 5, with 1 being the highest priority). The Computer Program Review Board (CRB) reviews defect data and directs corrective action. Direction may be to approve correction in the current baseline, defer correction to a future delivered baseline, forward to GE for correction in a future baseline, or withdraw a defect. Defects withdrawn from consideration might result from hardware problems, misunderstandings by personnel reporting defects, or other reasons.

The impact of upgrade items on the reliability of the delivered product is of great interest to NSWC as well as GE. NSWC specifies upgrade items for each AEGIS baseline. The cost of implementing upgrades in AEGIS weapons system software is specified in each baseline contract. Additional cost results from defect detection and correction in the delivered AEGIS weapons system software. NSWC would greatly benefit from having information on the reliability of each delivered baseline based on upgrade information. GE is interested in the reliability of delivered AEGIS software in order to improve the perfective maintenance process. If upgrade items and specification changes adversely impact reliability, improving the perfective maintenance process should result in improved reliability. GE desires feedback from NSWC to indicate where process improvements should be implemented, based on identification of the factors significantly affecting post-delivery product reliability.

The goal of analyzing defects recorded by NSWC is to determine if upgrade items and specification changes have significant relationship to the distribution and number of defects detected in the delivered AEGIS software product. If a significant relationship exists, quantification of the relationship must be produced.

The relationship between upgrade items, upgrade specification changes and post-delivery defects detected per functional group within the command and decision element was investigated. Specifically, the impact of upgrade items and upgrade specification changes on the reliability of functional groups within the control program was analyzed. Reliability in this context means the error-proneness of functional groups and was measured by defects detected per functional group.

The number of defects detected per functional group for Baseline 4 Phase II was captured using the NSWC database. Defects are detected throughout the life of the baseline. The data used in this analysis was collected on August 15, 1992, a full nine months after Baseline 4 Phase II was delivered. The number of upgrade items and upgrade specification changes affecting each functional group was collected at the time of delivery. The percentage of the module changed and the number of persondays expended per module to implement upgrade specification changes were also captured.

Simple graphs were constructed to compare the impact of perfective maintenance to the reliability of functional groups. The graphs indicated nonlinear relationships existed, suggesting rank order correlation be applied.

Rank correlation was applied to determine the association between the impact of perfective maintenance and the number of defects detected per functional group. Impact on functional groups is measured by the total number of times a functional group was changed to implement upgrade items, the total number of times a functional group was changed to implement specification changes, the number of persondays expended implementing upgrade items and specification changes, and the number of SLOCS added, deleted and changed to implement upgrade items and specification changes. Strong rank correlation existed between defects detected per functional group and perfective maintenance impact, as shown by the rank correlation coefficients in Table 6.5.

Table 6.5 Correlation Between Defects Detected and Perfective Maintenance Impact

	Total Number of Upgrades	Total Number of Upgrade Spec Changes	Total Number of SLOCs Added	Number of Persondays
Number of Defects Detected	.916	.902	.939	.838

The rank correlation coefficient shows significant correlation exists between defects detected and perfective maintenance impact. Strong correlation suggests functional groups can be ranked using maintenance impact. This ranking predicts the ranking of functional groups based on defects detected post-delivery.

Both NSWC and GE consider these results valuable, although for different reasons. NSWC can use the ranking to plan maintenance efforts based on functional groups significantly impacted by perfective maintenance efforts. GE can use the ranking of functional groups based on maintenance impact to select functional groups for additional testing in the Element Integration and Test phase. The relationship is also used during the PDR and CDR to identify functional groups with potential reliability problems.

Process Analysis

A large amount of data from the perfective maintenance process was collected throughout the Baseline 4 Phase II maintenance effort. The majority of this data was collected about the largest element computer program in the AEGIS weapons system, the command and decision element. Data was collected during the baseline effort by the project management organization, the software subcontractor, the metrics group, testing groups and upper level management. Validation was done by cross referencing data, performing an automated analysis of the delivered source code, and interviewing personnel.

The process data captured during Baseline 4 Phase II is of several different types, namely:

- the effort projected and actually expended on upgrade items and specification changes
- testing results
- the effort expended per month and per upgrade item
- computer program change per unit of effort (person day and person month).

Validated process data is stored and analyzed using the Exstatixs statistical analysis software package. Process data and product data are stored within the same database, allowing a large number of possible relationships to be statistically tested.

Upgrade Item Effort

Thirteen upgrade items were defined in the contract for the command and decision element program for Baseline 4 Phase II. The contract also contains an estimate of the effort, in personmonths, needed to implement each upgrade item. The estimates are produced using a GE proprietary procedure.

Table 6.6 shows the contract estimates of the personmonths needed to implement each upgrade item as a percentage of the actual personmonths expended. For example, the estimate of the number of personmonths needed to implement upgrade item number 3 was 48.04% of the actual personmonths expended. In this instance the estimate represents less than half the actual personmonths needed to implement the upgrade.

Table 6.6 Upgrade Item Effort Estimates as a Percentage of Actual

Upgrade Item	Estimate of Person Months as a Percentage of Actual Person Months
1	43.36
2	27.05
3	48.04
4	53.13
5	53.21
6	252.56
7	101.37
8	46.51
9	43.31
10	106.75
11	15.59
12	22.37
13	14.61

The contract estimates for each upgrade item need to be revised during the maintenance process using data from the current baseline and historical data from previous baseline efforts. As each baseline effort progresses, data becomes available about the progress and

status of the current effort. This data, used in conjunction with historical information, could provide valuable information about the current baseline effort.

One goal of the analysis of upgrade items was to determine if the actual effort required to implement an upgrade item could be predicted at two critical points in the maintenance process: immediately following PDR and imediately following CDR. If predictions could be made, the accuracy and timeliness must also be determined.

Applying the statistical technique of multiple linear regression, two predictive equations were derived, the first using data available at the PDR and the second using data available at the CDR. Table 6.7 presents the results obtained using multiple regression for both of these sets of data. The data items included in the regression equations were chosen using stepwise regression techniques which select the most statistically significant independent variables from the set of available independent variables, which included estimated computer program impact, estimated implementation effort, and estimated project data.

Table 6.7 Results of Upgrade Regression Analysis

Prediction Time	Dependent Variable	Independent Variables	R^2	P-value
PDR	Actual Person Months	• Contract Estimate of Person Months • Number of PDR Spec Changes	.82	.0011
CDR	Actual Person Months	• Contract Estimate of Person Months • Number of PDR Spec Changes • CDR Estimate of Changed SLOCs	.91	.0001

The prediction equations include three independent variables, two available at PDR and the third available at CDR. The estimate of personmonths in the contract and the number of specification changes approved at the PDR for each upgrade item are used in the PDR prediction equation. These same independent variables and the estimate of source lines of code (SLOCs) needed to be changed to implement each upgrade are included in the CDR

prediction equation. The dependent variable in both cases is the actual number of personmonths needed to implement an upgrade item.

Analysis of the prediction errors (residuals) was performed to determine if prediction intervals could be constructed. The residuals of each regression equation appear normally distributed. The residuals of each equation were divided at the median predicted value. The variance of residuals from above the mean and below the mean were found to be statistically equivalent for both equations. These assumptions are necessary to construct prediction intervals around the regression equations. Such prediction intervals have been constructed, but are not presented here for proprietary reasons.

The ability to construct prediction intervals increases organizational confidence in the regression equations. Prediction intervals are in use at this time. Project management uses the intervals as a window of variability of actual upgrade item effort.

Upgrade Specification Change Productivity

The defined process used to maintain computer programs allows coding to begin following the CDR. Theoretically, the requirements are stable before coding begins. In reality requirements continue to evolve and change throughout the maintenance effort. Evolution and change of requirements are documented as specification changes.

Post-CDR specification changes are recognized throughout the organization as having a significant impact on both the process and the product. There were more than twice as many specification changes after the CDR for the analyzed baseline than at the CDR and PDR. Post-CDR specification changes must be distributed to, and their implementation coordinated among, many different groups (i.e., system engineering, project management, subcontractors, quality assurance, testing groups and configuration management).

The goal of analyzing upgrade specification changes was to quantitatively evaluate their impact on both software products and the maintenance process. Specifically, the analysis sought to determine if PDR and CDR upgrade specification changes differed in process and product impact from post-CDR upgrade specification changes. If differences existed, quantification of these differences was required.

Ninety-one upgrade specification changes were written during the baseline. These specification changes were separated by origination time period as PDR, CDR or post-CDR. The characteristics of each group are shown in Table 6.8.

Table 6.8 Specification Change Characteristics by Origination Period

Origination Period	Number of Spec. Changes	Avg. Changed SLOCs per Spec. Change	Avg. Modules Changed per Spec. Change	Avg. Changed SLOCs per Module per Spec. Change	Productivity as a Percentage of CDR Productivity
PDR	13	1371.5	7.2	342.6	99.5%
CDR	17	1012.2	9.7	137.1	100%
Post CDR	61	84.4	1.8	44.9	72.4%

Two significant observations were made from Table 6.8, namely:

1. The productivity (measured by SLOCs/Person day) realized implementing post-CDR upgrade specification changes was 72.4% of the CDR specification change productivity.
2. Post-CDR specification changes have less impact on the computer programs, as measured by the average SLOCs changed, average modules changed, and average changed SLOCs per module, than PDR and CDR specification changes.

The Wilcoxon (also called the Mann-Whitney) rank sum test determined that a statistically significant difference existed between the average SLOCs changed per personday, SLOCs changed and SLOCs changed per module for post-CDR specification changes, and the same data for PDR and CDR specification changes. The data suggests that the size of the change might be responsible for the observed decrease in productivity.

Contingency tables were constructed to analyze the association between changed SLOCs and productivity for all upgrade specification changes. The tables were created using the median values of changed SLOCs per person day, changed SLOCs, and changed SLOCs per module to classify the data. The contingency tables are shown in Tables 6.9 and 6.10. The P-values for these tables are less than 0.005 for both tables allowing the

rejection of the hypothesis that changed SLOCs per person day is independent of changed SLOCs and changed SLOCs per module.

Table 6.9 Changed SLOCs per Personday vs. Changed SLOCs

	<= Median Changed SLOCs	> Median Changed SLOCs
<= Median SLOCs Changed per Personday	33	13
> Median SLOCs Changed per Personday	4	41

Table 6.10 Changed SLOCs per Personday vs. Changed SLOCs per Module

	<= Median Changed SLOCs per Module	> Median Changed SLOCs per Module
<= Median SLOCs Changed per Personday	42	3
> Median SLOCs Changed per Personday	4	42

Based on the information in Tables 6.9 and 6.10, and an analysis of the organization's process, it appeared that the overhead required to process and document specification changes, which had small impact on the control program, significantly decreased the productivity. A regression equation predicting persondays from changed SLOCs was fitted using all upgrade specification changes. The R^2 of this equation is .94 and the P-value is less than 0.0001. The value of b in this equation, representing the expected number of persondays added to each specification change regardless of the number of lines of code changed, was 6.1. Given the slope of the regression line is strictly positive, and that data points existed with zero changed SLOCs , the value of b indicates that each specification change requires a significant amount of overhead. This overhead decreases the productivity of specification changes which require a small number of changed SLOCs. Quantifying the overhead incurred by major activities in the maintenance process is an important step to understanding and managing the process. While small specification changes must be performed, an organization must account for this overhead to effectively assess, plan and perform software maintenance.

Engineering Test Phase

Weapons systems programs are written by a software subcontractor then delivered to the contracting organization. The delivered programs undergo two different test phases following delivery. The first test phase is the engineering test phase in which upgrade items are tested to detect errors and insure the specified functionality is achieved. The second test phase qualifies the entire system for demonstration to the customer.

The engineering test phase is an important phase in the maintenance process. It is the first phase which provides visibility to the customer of the ongoing maintenance effort. The number of errors found and the number of upgrades successfully completing the engineering test phase are documented and reported to the customer.

The goal of analyzing the engineering test phase was to determine the characteristics of upgrade items which influence the engineering test results, namely the number of errors detected and the testing failure rate. The influence of the characteristics was to be quantitatively specified.

Rank correlation and linear correlation analysis were applied to determine the relationship between upgrade item characteristics and engineering test results for the thirteen upgrade items. The correlation of the number of tests failed, number of tests run, and number of errors found are shown in Table 6.11. The correlation of upgrade item characteristics with either tests failed or errors detected is very low. The correlation of total tests run is statistically significant, suggesting a relationship exists between the upgrade characteristics of modules changed, persondays, SLOCs changed and total tests.

Further investigation of the engineering test phase showed very low first test failure rates for all upgrade items affecting the control program for the baseline under analysis. Both historical data and management input supported the view that first-test failure rates are typically much higher for the control program. The failure rates for the baseline under investigation are shown in Figure 6.1.

Table 6.11 Engineering Test Phase Result Correlation

Upgrade Item Characteristics	Rank Correlation			Linear Correlation
	Tests Failed	Errors Detected	Total Tests	Total Tests
Modules Changed	-0.031	0.439	0.594	0.633
Person Days	0.087	0.152	0.703	0.737
Spec. Changes	-0.366	0.084	0.257	0.027
SLOCs Changed	0.207	0.495	0.740	0.860

Analysis of the organization's maintenance process provided the key information needed to explain the quantitative results of Table 6.11 and Figure 6.1. In an effort to detect errors earlier in the maintenance process, the engineering test group supplied the software subcontractor with the tests to be executed during the engineering test phase. The software subcontractor used these tests to verify the computer program before the engineering test phase began.

Quantitative analysis of the subcontractor's test phase supports the assertion that errors were detected earlier in the baseline effort. The rank correlation between the number of upgrades and errors detected in the subcontractor test phase is .72. This correlation suggests the subcontractor detected and repaired the errors uncovered using the tests supplied by GE.

The number of errors detected during the engineering test phase does not correlate with any upgrade item characteristic. This is explained by the fact that the tests run during the engineering test phase have been run by the subcontractor in the immediately preceding phase of the maintenance process. The duplication of testing is expensive and unneeded. The organization is now in the process of changing the role of the engineering test group and the activities performed during the engineering test phase. The new focus of engineering test activities will be baseline to baseline regression. That is, the engineering test group will concentrate on testing the current baseline with tests from previous baselines to insure that no functionality is lost.

Figure 6.1 Percentage of First Tests Failed

Summary

This chapter described the results of applying statistical analysis techniques to process and product data collected throughout Baseline 4 Phase II of the AEGIS Weapons System. The first section presented significant product analysis results in three areas: module impact, incomplete computer program changes, and post-delivery product impact. Statistical analysis results of the GE perfective maintenance process were described in the second section. Significant statistical results in the areas of upgrade item effort, specification change productivity, and the effectiveness of the Engineering Test and Evaluation phase were presented. Chapter VII discusses of the validity and usefullness of the analysis techniques and the results presented in this chapter.

Chapter VII. Conclusions and Future Research

Introduction

This chapter summarizes the results of implementing the integrated assessment method described in Chapter III. Answers to the questions proposed in Chapter I are summarized, and the acheivement of the research objectives are discussed in the first section. The second section outlines the conclusions drawn from this research effort. The last section describes directions for future work.

Summary of Research Results

This research effort proposed and implemented an integrated approach for assessing the process used to develop and maintain software. The integrated approach described here has four major objectives:

1. To discover the activities comprising an organization's software process and the environmental factors affecting the process
2. To graphically document the process in a form usable by software personnel and amenable to analysis
3. To gather quantitative process and product data
4. To analyze an organization's process by considering activities present and absent within the process as well as statistical relationships between and among quantitative data

In order to achieve these four objectives, six significant questions were posed. These questions are

1. **Can the activities within an organization's software process be discovered?**
   Solution : This question relates to Objective 1. The activities within an organization's software process were discovered using the investigation method described in Chapter III. The investigation results are presented in the process description sections of Chapter V.

2. **Can the historical, environmental, and organizational factors impacting an organization's software process be determined?**
   Solution : This question relates to Objective 1. The historical, environmental, and organizational factors impacting an organization's software process were determined using the investigation method described in Chapter III. The historical, environmental, and organizational factor are presented in Chapter IV and Chapter V. Investigation results are discussed in the process analysis sections of Chapter V.

3. **Can an organization's software process be documented in a way usable to the organization and amenable to instrumentation?**

   Solution : This question relates to Objective 2. The GE software process and the perfective maintenance process were documented using the modeling technique presented in Chapter III. The GE software process model is not presented here due to proprietary concerns. The perfective maintenance process is graphically presented and described in Chapter V. The Specification Change Process, a subprocess of the perfective maintenance process, is attached in Appendix B.

4. **Can an organization's process be instrumented to obtain accurate process and product data?**

   Solution : This question relates to Objective 3. The GE software process was instrumented by carefully considering the investigation results and the process model, and by precisely specifying the data to be captured. The measurement techniques are presented in Chapter III. A large amount of process and product data was obtained and is described, where it is used, in Chapter VI.

5. **Can the content of an organization's software process be effectively analyzed?**

   Solution : This question relates to Objective 4. The content of the GE software process and the perfective maintenance process was analyzed using the techniques described in Chapter III. Content analysis results for both the GE software process and the perfective maintenance process are presented in Chapter V.

6. **Can statistically valid relationships between and among process data and product data be discovered and used to analyze an organization's software process?**

   Solution : This question relates to Objective 4. Statistically valid relationships between and among process data and product data were discovered using the statistical techniques described in Chapter III. Significant results in six areas were presented in Chapter VI.

Conclusions

Significant conclusions can be drawn from this research is several different areas, namely

1. The integration of the SEI assessment method and the principles of TQM result in a powerful software process assessment methodology. This research effort would have been far less successful without the use of both approaches. The use of SEI process content analysis and statistical analysis techniques combine to form an extremely effective assessment methodology.

2. The ability to capture information about the organizantion's historical, environmental and organizational factors provides important insight into why an organization's software process contains various activities. Simple adherence to the SEI approach would have caused this assessment to ignore the unique historical and organizational factors impacting the GE software process.

3. Software process assessment is a large task requiring continuous efforts. The success of this research effort was only possible through the the full year spent on-site. The likelihood of an SEI assessment group achieving the significant results described here while spending only two weeks on site, twice a year, is very unlikely.

4. Gathering simple process and product data throughout an organization's software process provides important insight into the effectiveness of the process and the impact of the software process on the software product. Insight is gained through statistical analysis of process and product data.

5. While assessment results in several areas suggested process improvements, other assessment results did not indicate the obvious improvements. Perhaps the adherence to Military Standard 1679 severely restricts the process improvements possible. More research is needed in this area.

Observations

Three important observations were made during the two year implementation of this research. Specifically,

1. The technical issues confronted and solved in this research, while significant and providing important results, were much easier than the cultural problems encountered. Changing the way an established, successful organization comprised of many professionals with lengthy experience is very much like turning a train; sharp rapid turns are impossible. Long slow changes of direction are the only possible options.

2. The automation of some portion of an organization's software process has a significant impact on employee job satisfaction. By removing the need for individuals to transfer paper forms and exchange infomation personally, much of enjoyment personnel derive from their jobs is lost. Many employees described increased automation as making them feel like part of a large, impersonal, machine. It is also very likely that important information communicated personally is lost using automation.

3. A large amount of diverse data about AEGIS software and the GE software process is collected by various groups within GE, the software subcontractor, and the customer auditing agency. Unfortunately, the information is not distributed to those who might use it and underutilized by those who maintain the data. No single group had access to all the data and therefore no one was able to gain an overall, quantitative view of the GE software process.

Future Research

The significant conclusions drawn from this research are based on a single, but very intensive, implementation of the integrated assessment methodology. In order to further test the effectiveness of the integrated assessment methodology, additional software organizations need be assessed.

A large amount of data was collected during the implementation phase of this research effort. Six areas of statistical analysis results are presented while many more areas are possible. In addition, several areas of statistical analysis should be more thoroughly investigated, namely the post-delivery product impact, testing effectiveness, and corrective maintenance activities.

Baseline-to-baseline data is not available at this time, severly restricting investigation of the use of assessment results across baselines. While process and product data was collected from Baseline 4 Phase II, little or no data remains from previous baselines. Data is being collected for baseline efforts currently underway but the project is not yet complete. Statistical analysis of data from different baseline efforts is needed to further validate the relationships discovered in this research effort.

The statistical analysis results presented in Chapter VI are significant but no description is given of the effect of process improvements on the statistical results. Further research is needed to specify the desired behavior of the statistical results when process improvements are implemented.

Bibliography

[ABDT90] Abdel-Hamid, T.K. and Madnick S. E., "Lessons Learned from Modeling the Dynamics of Software Development," *Communication of the ACM*, Vol. 32, No. 12, December 1990.

[ARTJ86] Arthur, J.D., Nance, R.E., and Henry, S.M., *A Procedural Approach to Evaluating Software Development Methodologies: The Foundation*, Technical Report SRC 86-008, Systems Research Center and Department of Computer Science Virginia Tech, 1986.

[ARTJ87] Arthur, J.D. and Nance, R.E., *Developing an Automated Procedure For Evaluating Software Development Methodologies and Associated Products*, Technical Report SRC 87-007, Systems Research Center and Department of Computer Science Virginia Tech, 1987.

[BASV87] Basili, V. R. and Rombach H. D., "Tailoring the Software Process to Project Goals and Environments," *Proceedings of the 9th International Conference on Software Engineering*, 1987.

[BLAR92] Blasewitz, R., personal communication, December 15, 1992.

[BOEB81] Boehm, B. W., *Software Engineering Economics*, Prentice Hall, 1981.

[BOEB91] Boehm, B. W., "Software Risk Management: Principles and Practices," *IEEE Software*, January 1991.

[BOLT91] Bollinger, T.B. and McGowen, C., "A Critical Look at Software Capability Evaluations," *IEEE Software*, July 1991.

[BOYJ91] Boyle, J., Executive Vice President, Video Lottery Consultants Inc., Bozeman Montana, personal communication, December 7, 1991.

[CARD89] Card, D.N., Berg, R.A., "An Industrial Engineering Approach to Software Development," *The Journal of Systems and Software*, October 1989.

[CHML90] Chmura, L. J., Norico, A. F., and Wicinski, T. J., "Evaluating Software Design Processes by Analyzing Change Data Over Time,"*IEEE Transactions on Software Engineering*, Vol. 16, No. 7, July 1990.

[CHRA87a] Chruscicki, A. J., "Software Reliability Assessment," Rome Air Development Center, Griffiss AFB, 1987.

[CHRA87b] Chruscicki, A. J., "Software Maintainability Prediction and Assessment," Rome Air Development Center, Griffiss AFB, 1987.

[COTV88] Cote, V., Bourque, P. and Rivard, N., "Software Metrics: An Overview of Recent Results," *The Journal of Systems and Software*, 1988.

[CRAS85] Crawford, S.G.and Fallah, M.H., "Software Development Process Audits - A General Procedure," *Proc. of the 8th International Conf. on Software Engineering*, August 1985.

[CURB87] Curtis, B. Krasner, H., Shen, V. and Iscoe, N., "On Building Software Process Models Under the Lamppost," *Proceedings of the 9th International Conference on Software Engineering*, 1987.

[DOD88] "Total Quality Management Master Plan," Department of Defense, August 1988.

[FAGM86] Fagan, M.E., "Advances in Software Inspections," *IEEE Transactions on Software Engineering*, Vol. 12, No. 7, July 1986.

[GESD91a] "AEGIS Computer Program Metrics Plan," General Electric Corporation, Government Electronic System Division, March 1991.

[GESD91b] "AEGIS Computer Program Development Standard," General Electric Corporation, Government Electronic System Division, December 1991.

[GRAR82] Grady, R. B. and Caswell, D. L., *Software Metrics: Establishing a Company-Wide Program*, Prentice Hall, 1987.

[HALM77] Halstead, M. H., *Operating and Programming Systems: Elements of Software Science*, Elsevier, 1977.

[HANG88] Hansend, G. A. and Kellner, M. I., "Software Process Modeling: The Triad Approach," *Proceedings of the 6th Symposium on Empirical Foundations of Information and Software Sciences*, October 1988.

[HENJ91a] Henry, J. E., Keenan, S., Keenan, M. and Henry, S., "An Assessment Method for Small Organizations," *Proceedings of the 1st International Software Quality Assurance Conference*, October 1991.

[HENJ91b] Henry, J. E. and Dublanica, W., "Improving the AEGIS Computer Program Development Process to Achieve SEI Maturity Level 3," General Electric Corporation, Government Electronic System Division, November 1991.

[HENJ92a] Henry, J. E., Dublanica, W., Kubeck, J. and McMullen, B., "A Process Modeling Technique and Application Results," *Proceedings of the Software Tools and Techniques Symposium*, March, 1992.

[HENJ92b] Henry, J. E.and Henry, S., "An Integrated Approach to Software Process Assessment," *Proceedings of the Pacific Northwest Software Quality Conference*, October 1992.

[HENJ92c] Henry, J. E.and Blasewitz, R. E., "Process Modeling: Theory and Reality," *IEEE Software*, November 1992.

[HENS90] Henry, S. and Selig, C., "Predicting Source Code Complexity at Design Time," *IEEE Software*, March 1990.

[HUFK86] Huff, K. E., Sroka, J. V. and Struble, D.D., "Quantitative models for managing software development processes," *Software Engineering Journal*, January 1986.

[HUMW87] Humphrey, W.S. and Sweet, W.L. "A Method for Assessing the Software Engineering Capability of Contractors," Software Engineering Institute, Carnegie Mellon University, September 1987.

[HUMW89a] Humphrey, W.S. and Kellner, M. I., "Software Process Modeling: Principles of Entity Process Models," *Proceedings of the 11th International Conference on Software Engineering*, May, 1989.

[HUMW89b] Humphrey, W.S., *Managing the Software Process*, Addison-Wesley, 1989.

[HUMW91a] Humphrey, W.S., "Software Process Improvement at Hughes," *IEEE Software*, July 1991.

[HUMW91b] Humphrey, W.S. and Curtis, B., "Comments on 'A Critical Look'," *IEEE Software*, July 1991.

[IMAR83] Iman, R. L., and Conover W. J., *Modern Business Statistics*, Wiley, 1983.

[JABJ91] Jablonski, J., R., *Implementing Total Quality Management: An Overview*, Pfeiffer, 1991.

[KAFD82] Kafura, D. and Henry, S., "Software Quality Based on Interconnectivity," *Journal of Systems and Software*, 1981.

[KATT89a] Katayama, T., "A Hierarchical and Functional Software Process Description and it's Enaction," *Proceedings of the 11th International Conference on Software Engineering*, May 1989.

[KATT89b] Katayama, T., "A Hierarchical and Functional Approach to Software Process Description," *Proceedings of the 4th International Software Process Workshop: Representing and Enacting the Software Process*, June 1989.

[KELM88] Kellner, M.I. and Hansen, G. A., "Software Process Modeling," Technical Report CMU/SEI-88-TR-9, Software Engineering Institute and Carnegie Mellon University, May 1988.

[KELM89a] Kellner, M.I., "Representation Formalisms for Software Process Modeling," *Proceedings of the 4th International Software Process Workshop: Representing and Enacting the Software Process*, June 1989.

[KELM89b] Kellner, M.I., "Experience with Enactable Software Process Models," *Proceedings of the 5th International Software Process Workshop: Experience with Software Process Models*, October 1989.

[KELM89c] Kellner, M.I., "Software Process Modeling Example," *Proceedings of the 5th International Software Process Workshop: Experience with Software Process Models*, October 1989.

[KELM89d] Kellner, M.I., "Software Process Modeling: Value and Experience," *SEI Technical Review*, Software Engineering Institute, Carnegie Mellon University, 1989.

[KELM90] Kellner, M.I., Session - Software Engineering Processes: Models and Analysis, *Proceedings of the 23rd Annual Hawaii International Conference on System Sciences, Vol. II - Software Track*, 1990.

[LANR90] Lanphar, R., "Quantitative Process Management in Software Engineering, A Reconciliation Between Process and Product Views,", *Journal of Systems and Software*, 1990.

[LEVY91] Levendel, Y., "Improving Quality with a Manufacturing Process," *IEEE Software*, March 1991.

[LEWK88] Lew, K. S., Dillon, T. S. and Forward, K. E., "Software Complexity and Its Impact on Software Reliability," *IEEE Transactions on Software Engineering*, November 1988.

[MCCT76] McCabe, T. J., "A Complexity Measure," *IEEE Transactions on Software Engineering*, May 1976.

[MCCT90] McCabe, T. J. and Butler, C. W., "Design Complexity Measurement and Testing," *Communications of the ACM*, Vol. 32, No. 12, December 1989.

[MOGJ90] Mogilensky, J., "Approaches to Upgrading Software Process Maturity," *Washington Ada Symposium*, June 1990.

[MYEG79] Myers, G. J., *The Art of Software Testing* , Wiley, 1979.

[OSTL87] Osterwell, S., "Software Processes are Software too," *Proceedings of the 9th International Conference on Software Engineering*, 1987.

[OTTL88] Ott, L., *An Introduction to Statistical Methods and Data Analysis*, PWS-Kent, 1988.

[QUIR80] Quinnan, R.E., " A programming process study," *IBM Systems Journal*, Vol. 19, No. 4, April 1980.

[ROSC87] Rosson, C. V., "Management Indicators: Assessing Product Reliability and Maintainability, Technical Reports SRC-88-011, Systems Research Center and Department of Computer Science Virginia Tech, 1988.

[SCHH88] Schultz, H.P., "Software Management Metrics," Technical Report ESD-TR-88-001, The Mitre Corporation, May 1988.

[SCHN87] Schneidewind, N. F., "The State of Software Maintenance," *IEEE Transactions on Software Engineering*, Vol. 13, No. 3, March 1987.

[SCHN92] Schneidewind, N. F., "Methodology for Validating Software Metrics," *IEEE Transactions on Software Engineering*, May 1992.

[SEI91] "Capability Maturity Model for Software," CMU/SEI-91-TR-24, Software Engineering Institute, Carnegie Mellon University, August 1991.

[SEIS92] Siegal, S., "Why We Need Checks and Balances to Assure Quality," *IEEE Software*, January 1992.

[SEL90] "Manager's Handbook for Software Development," Software Engineering Laboratory, Goddard Space Flight Center, November 1990.

[SHEV85] Shen, V.Y., Tze-Jie, Y., Thebaut, S.M. and Paulsen, L.R., "Identifying Error-Prone Software - An Empirical Study," *IEEE Transactions on Software Engineering*, Vol. 11, No. 4, April 1985.

[THAR82] Thayer, R. H., Pyster, P. and Wood, R. C., "Validating Solutions to Major Problems in Software Engineering Project Management," *IEEE Computer*, August 1982.

[TZEY88] Tze-Jie, Y., Shen, V.Y. and Dunsmore, H.E., "An Analysis of Several Software Defect Models," *IEEE Transactions on Software Engineering*, Vol. 14, No. 9, September 1988.

[WEID79] Weiss, D.M., "Evaluating Software Development by Error Analysis: The Data from the Architecture Research Facility," *Journal of Systems and Software*, No. 1, 1979.

[WEID85] Weiss, D.M. and Basili, V.R., "Evaluation of Software Development by Analysis of Changes: Some Data from the Software Engineering Laboratory," *IEEE Transactions on Software Engineering*, Vol. 11, No. 2, February 1985.

[WILL88] Williams, L. G., "Software Process Modeling: A Behavioral Approach," *Proceedings of the 10th International Conference on Software Engineering*, 1988.

Appendix A Model Generation Rules

The modeling technique contains specific definitions, symbology, generation rules applicable to all tiers and generation rules governing each specific tier.

A standard set of terms defines the information added to control flow diagrams. The terms and their definitions are as follows:

Inputs	Products delivered at the start of a phase from a previous phase or from external sources. The products are required to initiate the current phase.
Outputs	Products generated by a phase which are inputs to a subsequent phase or are necessary documentation at Tier 1.
Feedback In	A change, constraint or other impact accepted by a phase which is generated by another phase at Tier 1.
Feedback Out	A change, constraint or other impact generated a phase which is accepted by another phase at Tier 1.
Measurements	Values obtained during a phase used to quantitatively evaluate the process, product or phase.

The symbology used in this modeling technique is shown in Figure A-1.

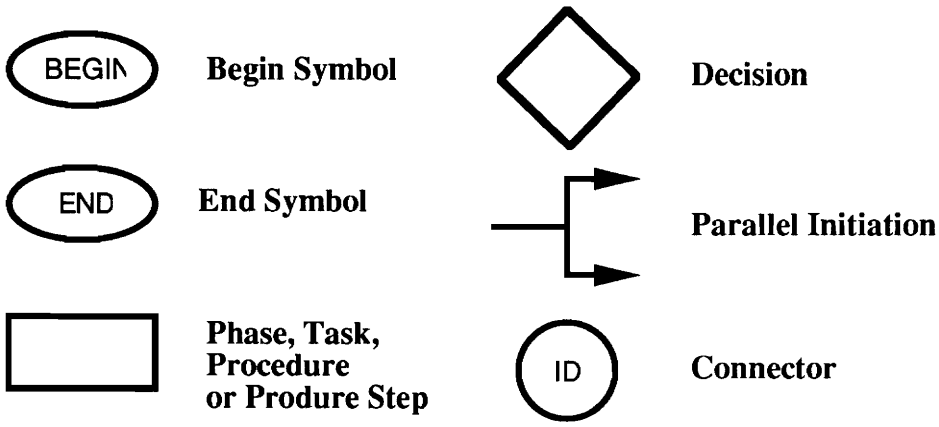

Figure A-1. Modeling Symbols

Eight generation rules govern construction of models through all tiers. These rules are:

1. All Output and Feedback Out shown on a tier 1 phase block shall have clearly identified generation blocks on tiers 2 and 3 of that phase.

2. All Inputs and Feedback In shown on a tier 1 phase block shall have clearly identified blocks accepting the Input or Feedback In on tiers 2 and 3 of that phase.

3. A block on any tier shall have exactly one control flow entrance point and one control flow exit point.

4. A decision block at any tier shall have exactly one control flow entrance point.

5. Blocks at all tiers shall be uniquely numbered. Tier 1 phases are numbered sequentially starting with 01. Blocks on Tiers 2 and 3 are numbered by adding a period and a number to the parent number of the block they describe. The following is an example of the numbering sequence for related blocks on Tiers 1 through 3.

   Tier 1: 01 - phase 01
   Tier 2: 01.1, 01.2, 01.3, 01.4 - the four tasks of phase 01
   Tier 3: 01.2.1, 01.2.2, 01.2.3 - the three procedures task 01.2

6. The single group within an organization responsible for a phase shall be stated in Tier 1. When the group performing a task, procedure, or procedure step is different from the group named in Tier 1, the description of the task , procedure or procedure step shall identify the responsible group.

7. Lines exiting a decision must go to a block or a different decision block.

8. All tiers shall start with a BEGIN symbol and terminate with an END symbol.

In addition to the general rules given above, specific generation rules for creating Tier 1 of a phase model are:

1. Tier 1 shall clearly depict each phase of the development process.

2. Tier 1 shall include for each phase the

   - Phase name and number,
   - Inputs,
   - Outputs,
   - Feedback In,
   - Feedback Out, and the
   - Responsible Group.

3. The phase is initiated upon receipt of the inputs.

The model generation rules for Tier 2 are:

1. Tier 2 shall provide an overview of the phase and should consist of four to ten tasks.

2. Each task shall have a single purpose and this purpose shall not overlap with any other task.

3. Tier 2 is initiated upon receipt of the inputs shown in Tier 1.

Specific generation rules for Tier 3 are:

1. Tier 3 shall be a depiction of the procedures needed to implement the tasks in Tier 2 and maintains the same control flow shown in Tier 2.

2. Each procedure block shall map to a single task shown on Tier 2.

3. The single control flow entrance point and exit point of each task shown in Tier 2 shall be maintained in Tier 3.

Appendix B Specification Change Process Standard

AEGIS Computer Program Standards
Specification Change Process Definition

1.0 DESCRIPTION

This standard establishes procedures for the generation, management and control of changes to Program Performance Specifications (PPS) and Interface Design Specifications (IDS), and the associated changes to computer programs for the AEGIS Weapons System Computer Elements Controlled by Naval Systems Department (NSD).

2.0 SCOPE

This standard applies to all AEGIS computer program development efforts, as specified in the AEGIS Computer Program Development Plan (CPDP).

3.0 RESPONSIBILITIES

NSD Systems Engineering owns the specification documents until implementation of computer programs begins. System Engineering generates Specification Changes (SCs) and Interface Change Requests (ICRs), and responds to Systems Definition Requests (SDRs). Systems Engineering verifies that developed computer programs satisfy system specifications.

The specification documents are owned by NSD Computer Program Development (CPD) once implementation begins. CPD evaluates computer program changes for cost and schedule impact. CPD directs and manages the implementation of computer program changes through out all development processes.

The software developer implements computer program changes described by SCs under the direction and control of CPD.

Configuration Projects participates in the preparation, review, approval, revision and general processing of specification documents and SCs with other groups. Configuration Projects maintains appropriate status records to assure specifications and SCs are complete and included in the specification documents. The preparation and distribution of Specification Change Notices (SCNs) are the responsibility of Configuration Projects.

4.0 REFERENCED/RELATED DOCUMENTS

NS-M-CMO-D-0024B Specification Change Process

NS-M-CMO-D-0048D Policy for the Preparation, Release and Control of Specifications

NS-M-CMO-D-0063A ACS Interface Control Documentation Procedures

NS-M-SCP-D-4020 Software Implementation Board (SIB)

NS-A-CMO-D-1009 Naval Systems Specification Identification and Control

NS-M-SCP-D-4001A Specification Change Review Implementation Baseline and Evaluation (SCRIBE) Board

NS-M-SCP-D-4018 Rules for Test Observation Reporting and Computer Program Change Request (CPCR) Implementation

NS-M-SCP-D-4019 Origination and Processing Test Observation Reports

NS-M-SCP-D-4020 Definition and Use of Specification Change Status Fields

5.0 SPECIFICATION CHANGE PROCESS

During the preparation of a specification document, the technical requirements of the system are established, documented and refined by the technical community responsible for the performance and interface characteristics of the system. Changes are typically required as the system requirements become better understood and progressively refined.

Specifications are established and reviewed during the development process. During this process the specification document is categorized as Advance Copy, Released Copy, or Issued Copy. The categories are defined as follows:

- **Advance Copy** - A specification document distributed to the cognizant engineering community for review, but whose contents are subject to revision and will not be released or placed under formal change control are termed Advance Copy. The document front page shall be marked "ADVANCE COPY - FOR REVIEW ONLY" by Engineering Documentation and may be used for Preliminary Design Review (PDR), Critical Design Review (CDR), and/or other In-Process Review (IPRs).

- **Released Copy** - A specification document sufficiently stable to be utilized as a design document by the entire technical community is termed Released Copy. The document front page shall be marked "RELEASED - UNDER GE CONFIGURATION CONTROL". Normally, Advance Copy specification documents shall be upgraded to formal CM controled Released Copies after PDR and/or CDR comments are incorporated. Specifications which evolve from earlier versions without major change may also be upgraded to Released Copy prior to the scheduled PDR or CDR. Subsequent changes shall be made by the SC process described in this standard.

- **Issued Copy** - A specification document complete from a technical and quality standpoint and submitted for delivery in accordance with contract requirements (DD-250 or contract letter) is termed Issued Copy. All Advance Copy, Released Copy and other temporary annotations shall be removed from the issued copy.

5.1 Generation of Specification Changes

Proposed changes to Advanced Copy specifications shall be jointly controlled by System Engineering and Computer Program Development. Changes to Released Copy and Issued Copy specifications are implemented under formal change control by the Configuration Projects. The proposed changes are typically the result of Baseline Functional Upgrades, Navy comments, Specification Definition Requests and ET type Computer Program Change Requests (CPCRs).

Baseline Functional Upgrades are contractually driven changes to existing computer programs. Systems Engineering shall, as directed by the prime contract, generate SCs to support functional changes to existing computer programs.

Members of the Element System Engineering group shall review and analyze Navy comments, Specification Definition Requests and ET type CPCRs. Representatives of the Element System Engineering group, Computer Program Development and the developer shall discuss problems discovered during this analysis and determine if a change to a specification is required. If no change is required System Engineering shall document a response and distribute the response to the originator and other personnel as required. If a change to the specification is required, the change shall be documented via a Specification Change Form (see attached).

5.2 Preparation and Evaluation of Specification Changes

Representatives of the Element System Engineering group, Computer Program Development, and the developer shall discuss all SCs. The specification document change pages and code impact for each SC shall be determined.

System Engineering shall be responsible for disclosing design concepts and document change pages. The change pages shall be attached to the Specification Change Form.

If an SC impacts the computer programs, Computer Program Development shall direct the developer to estimate the effort required to implement an SC and the impact on computer programs. Advance copies of all SCs impacting computer programs shall be forwarded to the developer configuration management group. The developer configuration management group shall forward SCs to the group within the developer organization responsible for implementing the impacted element. The element developer shall prepare an estimate of the effort required to implement each SC. The estimate shall include man-days required, the lines of code to be added, changed and deleted, the number of cells to be added, changed and deleted, and the program modules affected.

Representatives of CPD shall review all SCs and associated estimates to determine the impact on cost and schedule. CPD shall discuss and resolve conflicts and open issues regarding SCs and associated estimates prior to designating SCs for review and direction. The origination (ET type CPCR, SDR, functional upgrade, maintenance, or other) of all SCs shall be recorded and maintained by CPD.

Computer Program Development shall designate SCs to be included in the PDR Package, the CDR Package, forwarded to the SCRIBE, or put on HOLD.

Computer Program Development shall insure that all SCs designated for review and direction are completely prepared and evaluated prior to PDR, CDR or SCRIBE meeting.

5.3 Review and Direction of Specification Changes

5.3.1 CDR and PDR Specification Changes

SCs written, evaluated and included in the PDR or CDR Package shall be reviewed and directed as part of the PDR or the CDR process. The decision to include an SC in the PDR or CDR Package shall be made by the Element Project Engineer. The ESTIMATE information described in Section 5.2 for each SC shall be included in the PDR or CDR Package.

SCs reviewed and directed at the PDR or the CDR shall be submitted to the SCRIBE secretary. The SCRIBE secretary shall forward SCs impacting computer programs to the SIB. The SIB secretary shall assign a unique Computer Program Change Request number to each SC which impacts the computer programs. Final SCRIBE minutes shall be published documenting the SCs reviewed and directed at the PDR and the CDR.

5.3.2 Post-Critical Design Review Specification Changes

A Specification Change written after the Critical Design Review shall be reviewed and directed by the SCRIBE. The Element Project Engineer shall designate SCs for review by the SCRIBE. SCs submitted to the SCRIBE shall be included in the SCRIBE agenda. The ESTIMATE information described in Section 5.2 shall be documented in the SCRIBE cover page. The SC and the SCRIBE cover page shall be submitted to the SCRIBE for review and direction at the next SCRIBE meeting.

The SCRIBE shall review SCs and assign one of the following statuses:

- Approve - The SC is approved for implementation
- Hold - The SC is held pending further information
- Withdraw - The SC is withdrawn from consideration

The Software Implementation Board shall generate a Computer Program Change Request for each SC approved by the SCRIBE which has impact on computer programs, unless the SC is in response to an existing ET type CPCR.

The SCRIBE evaluation status of each Change submitted to the SCRIBE shall be documented in the Preliminary SCRIBE minutes. The Preliminary SCRIBE minutes shall be published and submitted to the Navy for review. The Navy shall have five days to review and comment on the Preliminary SCRIBE minutes. If the Navy has no concerns, Final SCRIBE minutes shall be published. Concerns expressed by the Navy shall be discussed and the SC re-evaluated. Re-evaluated SCs shall be noted in the Final SCRIBE minutes.

5.4 Management and Control of Implementation

Computer Program Development shall direct the developer to implement PDR, CDR and SCRIBE approved SCs impacting computer programs. Management and control of the implementation of SCs shall adhere to NS-M-SCP-D-4018, Rules for Test Observations Reporting and Computer Program Change Request (CPCR) Implementation.

6.0 Tracking and Reporting of Specification Changes

STARSYS5, in the near future, shall support tracking evaluation, review, direction and computer program impact of SCs. STARSYS5 shall maintain a database of SC information including the fields shown on page one of the Specification Change Form and additional fields detailing evaluation, review, direction and implementation data. STARSYS5 shall implement and enforce the use of the SC status fields as detailed in NS-M-SCP-D-XX, Definition and Use of SC Status Fields.

Appendix C SEI Questionaire

1 Organization and Resource Management

1.1 Organizational Structure

1.1.1 For each project involving software development, is there a designated software manager?

1.1.2 Does the project software manager report directly to the project (or project development) manager?

1.1.3 Does the Software Quality Assure (SQA) function have a management report channel separate from the software development project management?

1.1.4 Is there a designated individual or team responsible for the control of software interfaces?

1.1.5 Is software system engineering represented on the system design team?

1.1.6 Is there a software configuration control function for each project that involves software development?

1.1.7 Is there a software engineering process group function?

1.2 Resources, Personnel and Training

1.2.1 Does each software developer have a private computer-supported work station/terminal?

1.2.2 Is there a required training program for all newly appointed development managers designed to familiarize them with software project management?

1.2.3 Is there a required software engineering training program for software developers?

1.2.4 Is there a required software engineering training program for the first-line supervisors of software development?

1.2.5 Is a formal training program required for design and code review leaders?

1.3 Technology Management

1.3.1 Is a mechanism used for maintaining awareness of the state-of-the -art in software engineering technology?

1.3.2 Is a mechanism used for evaluating technologies used by the organization versus those externally available?

1.3.3 Is a mechanism used for deciding when to insert new technology into the development process?

1.3.4 Is a mechanism used for managing and supporting the introduction of new technologies?

1.3.5 Is a mechanism used for identifying and replacing obsolete technologies?

2 Software Engineering Process and its Management

2.1 Software Engineering Process and its Management

2.1.1 Does the software organization use a standardized and documented software development process on each project?

2.1.2 Does the standard software development process documentation describe the use of tools and techniques?

2.1.3 Is a formal procedure used in the management review of each software development prior to making contractual commitments?

2.1.4 Is a formal procedure used to assure periodic management review of the status of each software development project?

2.1.5 Is there a mechanism for assuring that software subcontractors, if any, follow a disciplined software development process?

2.1.6 Are standards used for the content of software development files/folders?

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

2.1.8 Is a mechanism used for assessing existing designs and code for reuse in new applications?

2.1.9 Are coding standards applied to each software development project?

2.1.10 Are standards applied to the preparation of unit test cases?

2.1.11 Are code maintainability standards applied?

2.1.12 Are internal design review standards applied?

2.1.13 Are code review standards applied?

2.1.14 Is a formal procedure used to make estimates of software size?

2.1.15 Is a formal procedure used to produce software development schedules?

2.1.16 Are formal procedures applied to estimating software development cost?

2.1.17 Is a mechanism used for ensuring that the software design teams understand each software requirement?

2.1.18 Are man-machine interface standards applied to each appropriate software development project?

2.2 Process Metrics

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

2.2.2 Are profiles of software size maintained for each software configuration item, over time?

2.2.3 Are statistics on software design errors gathered?

2.2.4 Are statistics on software code and test errors gathered?

2.2.5 Are design errors projected and compared to actuals?

2.2.6 Are code and test errors projected and compared to actuals?

2.2.7 Are profiles maintained of actual versus planned software units designed, over time?

2.2.8 Are profiles maintained of actual versus planned software units completing unit testing over time?

2.2.9 Are profiles maintained of actual versus planned software units integrated over time?

2.2.10 Are target computer memory utilization estimates and actuals tracked?

2.2.11 Are target computer throughput utilization estimates and actuals tracked?

2.2.12 Is target computer I/O channel utilization tracked?

2.2.13 Are design and code review coverages measured and recorded?

2.2.14 Is test coverage measured and recorded for each phase of functional testing?

2.2.15 Are the action items resulting from design reviews tracked to closure?

2.2.16 Are software trouble reports resulting from testing tracked to closure?

2.2.17 Are the action items resulting from code review tracked to closure?

2.2.18 Is test progress tracked by deliverable software component and compared to the plan?

2.2.19 Are profiles maintained of software build/release content versus time?

2.3 Data Management and Analysis

2.3.1 Has a managed and controlled process database been established for process metrics data across all projects?

2.3.2 Are the review data gathered during design reviews analyzed?

2.3.3 Is the error data from code reviews and tests analyzed to determine the likely distribution and characteristics of the errors remaining in the product?

2.3.4 Are analyses of errors conducted to determine their process related causes?

2.3.5 Is a mechanism used for error cause analysis?

2.3.6 Are the error causes reviewed to determine the process changes required to prevent them?

2.3.7 Is a mechanism used for initiating error prevention actions?

2.3.8 Is review efficiency analyzed for each project?

2.3.9 Is software productivity analyzed for major process steps?

2.4 Process Control

2.4.1 Does senior management have a mechanism for the regular review of the status of software development projects.

2.4.2 Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?

2.4.3 Is a mechanism used for identifying and resolving system engineering issues that affect software?

2.4.4 Is a mechanism used for independently calling integration and test issues to the attention of the project manager?

2.4.5 Is a mechanism used for regular technical interchanges with the customer?

2.4.6 Is a mechanism used for ensuring compliance with the software engineering standards?

2.4.7 Do software development first-line managers sign off on their schedules and cost estimates?

2.4.8 Is a mechanism used for ensuring traceability between the software requirements and top-level design?

2.4.9 Is a mechanism used for controlling changes to the software requirements?

2.4.10 Is there a formal management process for determining if the prototyping of software function is an appropriate part of the design process?

2.4.11 Is a mechanism used for ensuring traceability between the software top-level and detailed designs?

2.4.12 Are internal software design reviews conducted?

2.4.13 Is a mechanism used for controlling changes to the software design?

2.4.14 Is a mechanism used for ensuring traceability between the software detailed design an the code?

2.4.15 Are formal records maintained of unit (module) development progress?

2.4.16 Are software code review conducted?

2.4.17 Is a mechanism used for controlling changes to the code? (Who can make changes and under what circumstances?)

2.4.18 Is a mechanism used for configuration management of the software tools used in the development process.

2.4.19 Is a mechanism used for verifying that the samples examined by Software Quality Assurance are truly representative of the work performed?

2.4.20 Is there a mechanism for assuring that regression testing is routinely performed?

2.4.21 Is there a mechanism for assuring the adequacy of regression testing?

2.4.22 Are formal test case reviews conducted?

3 Tools and Technology

3.1 Is automated configuration control used to control and track change activity throughout the software development process?

3.2 Are computer tools used to assist in tracing software requirements to software design?

3.3 Are formal design notations such as PDL used in program design?

3.4 Are computer tools used to assist in tracing the software design to the code?

3.5 Is the majority of prod

3.6 Are automated test input data generators used for testing?

3.7 Are computer tools used to measure test coverage?

3.8 Are computer tools used to track every required function and assure that it is tested/verified?

3.9 Are automated tools used to analyze the size and change activity in software components?

3.10 Are automated tools to analyze complexity?

3.11 Are automated tools used to analyze cross references between modules?

3.12 Are interactive source-level debuggers used?

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

3.14 Are computer tools used for tracking and reporting the status of the software in the software development library?

3.15 Are prototyping methods used in designing the critical performance elements of the software?

3.16 Are prototyping methods used in designing the critical elements of the man-machine interface?

Appendix D Perfective Maintenance Follow-up Questionaire

Interviewee : _____

Interviewer : _____

General The purpose of this section is to gather information about the SC process in general.

Could you describe your view of the SC process?

Where do SCs originate?

Who gets copies of them?

What information is gathered about SCs?

What happens to SCRIBE approved SCs?

SCRIBE differed?

SCRIBE withdrawn?

Who is affected by approved SCs?

What documentation is generated about approved SCs?

Who writes SDRs and when?

Specific The purpose of this section is to gather information about how the SC process directly involves the person interviewed.

What is your role in the SC process?

What information about SCs do you provide?

What information about SCs do you use?

What decisions about SCs do you make?

What type of SC reports do you use?

Who generates the reports?

Where does the data for these reports reside?

Who gathers and maintains the data?

What is the biggest problem for you involving SCs?

What change or improvement would help you most involving SCs?

Vita of

JOEL E. HENRY

223 W. Pomfret St.
Carlisle, PA 17013
717-258-5602

Education:

Ph.D. candidate in Computer Science at Virginia Polytechnic Institute and State University, Blacksburg, VA.

MS (1986) in Computer Science from Montana State University, Bozeman, MT.

BS (1985) in Computer Science from Montana State University, Bozeman, MT.

Employment:

Instructor of Computer Science (8/92 - present)
Department of Mathematics and Computer Science
Dickinson College
Carlisle, PA 17013

Technical Consultant (5/91 - 11/91, 4/92 - 8/92)
Government Electronic Systems Division
General Electric
Borton Landing Rd.
Moorestown, NJ 08057

Assistant Professor of Computer Science (8/86 - 5/89)
Computer Science Department
Daniel Webster College
Nashua, NH 03063

Software Engineer (5/88 - 11/88)
Artificial Intelligence Technical Center
Digital Equipment Corp.
Marlboro, MA

Programmer/Analyst (5/85 - 8/86)
Center for Data Systems and Analysis
Montana State University
Bozeman, MT 59715

Assistant Manager (6/82 - 9/84)
Frederick's Restaurant
117 North Seventh Ave.
Bozeman, MT 59715