Technical Report
CMU/SEI-93-TR-07
ESC-TR-93-184

Carnegie-Mellon University
Software Engineering Institute

(1)

The SEI and NAWC: Working Together to Establish a Software Measurement Program

James A. Rozum

December 1993

Technical Report
CMU/SEI-93-TR-07
ESC-TR-93-184
December 1993

The SEI and NAWC: Working Together to Establish a Software Measurement Program

James A. Rozum
Software Process Measurement Project

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

SEI Joint Program Office
ESC/ENS
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

[illegible]

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

The *Software Engineering Institute* is sponsored by the U.S. Department of Defense.

This report was funded by the U.S. Department of Defense.

Table of Contents

Preface v

Acknowledgments vii

1. Introduction and Background 1
- 1.1 Purpose and Audience 1
- 1.2 Report Organization 1
- 1.3 NAWC - Aircraft Division, Warminster, Pa. 2
- 1.4 Beginning A Measurement Program at NAWC 2
- 1.5 Criteria for the NAWC Measurement Process 3

2. Development Strategy for the Measurement Process 5
- 2.1 Software Measurement Process Action Team 5
- 2.2 SMPAT Action Plan and Implementation Strategy 6
  - 2.2.1 Phase 1: Exploration and Planning 7
  - 2.2.2 Phase 2: Establishing Key Management Concerns 8
  - 2.2.3 Phase 3: Defining Measurement Data 8
  - 2.2.4 Phase 4: Defining Data Collection and Analysis Processes 8
  - 2.2.5 Phase 5: Pilot Testing the Measurement Process 9
  - 2.2.6 Phase 6: Conducting Software Measurement Training and Awareness Briefings 10
- 2.3 Identify Management Issues and Concerns 10
  - 2.3.1 Resource Issues 11
  - 2.3.2 Progress Issues 11
  - 2.3.3 Quality Issues 11
  - 2.3.4 Pre-Contract Award Issues 12
- 2.4 Description of Development Increments 13
  - 2.4.1 Description of Increment Two Measures 15
  - 2.4.2 Description of Increment Three Measures 16
  - 2.4.3 Description of Increment Four Measures 17

3. Description of Increment One Measurement Processes 19
- 3.1 Issues to Address 19
- 3.2 Data Definitions 20
- 3.3 Data Collection Process 30
  - 3.3.1 Collection of Staffing Data 31
  - 3.3.2 Collection of Effort Data 31
  - 3.3.3 Collection of Product Delivery (Milestones) Data 32
  - 3.3.4 Collection of Work Progress Data 32
  - 3.3.5 Collection of Size Data 32
  - 3.3.6 Collection of Post Deployment Work Data 33
- 3.4 Data Analysis Process 33
  - 3.4.1 Resource Issue Analyses 34
  - 3.4.2 Progress Issue Analyses 43
  - 3.4.3 Analyses of Technical Quality Issues 51
- 3.5 Reporting the Analysis Results 56

4. Piloting the Processes 57

5. Software Metrics Office and Database 61
5.1 Software Metrics Office 61
5.2 Measurement Database 62

6. Lessons Learned 65

7. Future Directions and Conclusions 69

Bibliography 71

Appendix A — Acronyms 73

Appendix B — Data Definition Forms 75

Appendix C — Data Collection Forms 83

List of Figures

Figure 2-1	Linkage of SMPAT to Other NAWC Activities	6
Figure 3-1	Sample Staff Availability	35
Figure 3-2	Sample Staff Stability	37
Figure 3-3	Sample Cumulative Staff-Hour Data	38
Figure 3-4	Sample Monthly Staff-Hour Data	39
Figure 3-5	Sample Effort Distribution Chart	41
Figure 3-6	Sample Effort Distribution by Functional Area	42
Figure 3-7	Sample CSCI Software Size Tracking	43
Figure 3-8	Sample Milestone Commitment and Planning	44
Figure 3-9	Sample Analysis Using Multiple Plans	45
Figure 3-10	Sample Indicator for CSU Development Progress	47
Figure 3-11	Sample Milestone Completion Variances	48
Figure 3-12	Sample Functionality Allocation Plan	49
Figure 3-13	Sample Backlog Report of SPRs	50
Figure 3-14	Sample Progress Delays Due to Open SPRs	51
Figure 3-15	Sample SPR Density	52
Figure 3-16	Sample Pareto Analysis of CSCI Defects	54
Figure 3-17	Sample SPR Longevity Report	55
Figure 4-1	Pilot Project Timeline	58
Figure 5-1	List of Basic Indicators Produced Monthly	63

Preface

The software measurement program at the Naval Air Warfare Center (NAWC) at Warminster, Pa., started in 1989 when the director of the Computer Systems and Software Department recognized the need to improve the management insight and control of software projects. With the guidance of Watts Humphrey and the SEI model for process improvement [PAULK], NAWC soon realized that software measurement was also an important part of all software processes.

In 1990, the NAWC at Warminster, Pa., with the help of the Software Engineering Institute (SEI), initiated a software measurement initiative. The intent of the measurement initiative was to provide quantitative methods that could be used to improve the management and control of software systems development and maintenance at NAWC-Warminster.

With assistance from the SEI, the initiative has completed a software measurement guide that details what specific measurement data to collect, how to collect the data, why the data are being collected, and how the data will be used. The staff members working on the initiative have also completed a measurement database, and pilot tested both a software measurement database and the methods detailed in the guide.

This technical report is intended to share some of the success from this SEI joint effort with the software community. The report describes how the effort was started and planned and how the first of four increments of measures were implemented. An organization that is about to start a measurement effort should find the level of detail in this report helpful in determining how to plan its program and focus its initial effort. The lessons learned included at the end of this report should also help organizations avoid some potential pitfalls.

Acknowledgments

Of course, a joint effort such as the one between the SEI and NAWC that is described in this technical report is the product of many individuals. Although there are too many to list all of the contributions here, two key contributors were Mr. Watts Humphrey of the SEI and Mr. Richard Mitchell of NAWC. Mr. Humphrey and Mr. Mitchell were the sponsors and supporters of this effort. It was Mr. Mitchell's commitment to improvement that made the entire effort possible. Another key contributor was Mr. Charles Koch of NAWC who was responsible for organizing the effort and implementing it at NAWC. He also chaired the software measurement process action team (SMPAT).

I thank my fellow SMPAT members and the contributions that they have made: Robert Jankiewicz, Helen Ling, Tom Nguyen, James Rachiele, and Mien Wann.

I also thank the contributions of those who reviewed early versions of this technical report and provided many useful comments to improve it: Anita Carleton, Wolfhart Goethert, Donald McAndrews, and Robert Park.

Special thanks go to Suzanne Couturiaux who did an outstanding job editing this technical report and helped to improve the early versions.

The SEI and NAWC: Working Together to Establish a Software Measurement Program

Abstract. In 1990, the Software Engineering Institute (SEI) and the then Naval Air Development Center (NADC) in Warminster, Pennsylvania (now Naval Air Warfare Center, Aircraft Division, Warminster) signed an agreement to jointly develop a software measurement program for NADC - Warminster. To help with that development, a software measurement process action team (SMPAT) was formed with members from the SEI and NADC. The SMPAT's responsibility was to plan, develop, and assist in the implementation of the measurement program. The purpose of this technical report is to document and make available to the software community the process and methods used, experience gained, and some lessons learned in establishing the software measurement program at NADC.

This report is meant to help organizations that desire to start a software measurement program or have been struggling with such a program by providing an example of one organization that has also struggled to establish a software measurement program. To help an organization, real-life examples of how software measures were defined, collected, and used to improve the management process are included.

1. Introduction and Background

1.1 Purpose and Audience

The purpose of this report is to document and make available to the software community the process and methods used, experience gained, and some lessons learned in establishing a software measurement program at the Naval Air Warfare Center. The audience for this report is potentially any software organization that has, or wants to establish, a software measurement program of its own. The intent of this report is to help those organizations that desire to start a software measurement program or have been struggling with such a program by providing an example of one organization that has also been struggling to establish a software measurement program. Through this real-life example, the SEI hope is that an organization will be able either to "jump start" its measurement program or guide itself through and around some of the obstacles in establishing a software measurement program.

1.2 Report Organization

The intent of this technical report is to give the complete process that was followed by the SEI and NAWC in establishing the software measurement program at NAWC. Chapter 1 gives background information and a description of NAWC. Chapter 2 discusses the strategy and plan to develop the measurement program. As part of that strategy, the measurement issues to be addressed by the measurement program were divided into four increments.

Chapter 3 discusses the processes and measures that were implemented as part of increment one, including:

- Complete data definitions for the software measures.
- Details on how and where the data were collected.
- Examples of how the NAWC metrics office might analyze the software data to address specific issues.
- Process of reporting the data analysis results back to the program manager.

Chapter 4 discusses the pilot test that was conducted to verify the processes in Chapter 3. Chapter 5 describes the role of the NAWC metrics office and the software measurement database developed. Chapter 6 discusses the lessons learned in establishing the measurement program at NAWC.

1.3 NAWC - Aircraft Division, Warminster, Pa.

The software measurement program at the Naval Air Warfare Center (NAWC) in Warminster, Pennsylvania started in 1989 when the director of the Computer Systems and Software Department recognized the need to improve the management insight and control of software projects. The Computer Systems and Software Department at NAWC-Warminster, or Code 70, is responsible for supplying computer and software technology to other projects. The Code 70 director realized that software measurement would also have to be an important part of all software processes.

NAWC-Warminster, formerly the Naval Air Development Center, is now part of the Naval Air Warfare Center (NAWC) that includes navy laboratories throughout the United States including: Pt. Magu and China Lake, California; Warminster, Pennsylvania; Patuxant River, Maryland; Indianapolis, Indiana; and several other smaller sites.

NAWC-Warminster (hereafter, NAWC-Warminster will be referred to as simply NAWC) is the primary navy laboratory responsible for anti-submarine warfare (ASW) subsystems and avionics applications. Much of NAWC's ASW work requires extensive computer support both during development and for the many electronic components embedded in ASW subsystems. Typical tasks at NAWC include technical management and performance of ASW system upgrades, new developments of avionics systems, and independent verification and validation (IV&V) of major naval aircraft system upgrades.

1.4 Beginning A Measurement Program at NAWC

In the quest to improve its software capability, NAWC realized it needed to establish a software measurement program. Since the objective was to improve the software process, it was essential to be able to measure improvement both of the process and of the products produced by the process. Without measurement, it would not be possible to determine objectively if a given process change is beneficial, detrimental, or has no effect. In examining the SEI Capability Maturity Model (CMM) and its associated assessments, NAWC has recognized the implicit requirement for software measurement to reach the repeatable level of the CMM. Therefore, NAWC established a project to examine the measurement possibilities, define a set of measures, and begin a measurement process on a pilot set of projects. This project, performed by the Software Engineering Technology Branch of the Computer Systems and Software Department, was to initiate software measurement activities to support planning, control, improvement, management visibility, and tracking of the software process at NAWC. The goal was to be able to provide insight—quantitatively—into the issues and questions that software managers have about a specific project. It will also provide a mechanism to monitor the continuing process of improvement in NAWC's software capability. In August 1990, NAWC and the SEI signed an agreement to work jointly toward improving, developing, and implementing a software measurement program at NAWC.

In 1991, a Naval Air Systems Command (NAVAIR) task force, including two NAWC personnel from Warminster, developed the NAVAIR Avionics Instruction for Metrics [NAVAIR]. That document serves as a policy statement that mandates the collection of software measurement data. The policy contains high-level descriptions of the measures and leaves the details (e.g., definitions, collection methods, analysis) to the individual projects. This serves the purpose of allowing maximum flexibility in implementing the software measurement, without imposing requirements that could have an impact on the way projects currently operate; it thus obviates large costs to implement the instruction. The software measurement program described in this report was NAWC's implementation of that policy at the laboratory level.

To help with that implementation, a software measurement process action team (SMPAT) was formed in July 1991, to develop and implement a software measurement program at NAWC.

The NAWC measurement program, to date, has been accomplished mostly on a "shoe-string" budget. The SMPAT chairperson was only available part-time to work on software measurement issues and was supported with minimal internal funding. The SMPAT members were primarily volunteers, attending meetings and reviewing documents as they could. NAWC has been able to provide a small amount of funding to the SEI for software measurement support. Recently, NAWC has been able to fund a full-time member of its staff to collect and analyze data on the pilot program and develop a database for measurement data. This one person represented the beginning of a software metrics office at NAWC [ROZUM93].

1.5 Criteria for the NAWC Measurement Process

NAWC realized early that the success of its measurement program would depend upon the program's acceptance by the projects that will implement, collect, and analyze the measures. To gain cooperation and acceptance, the NAWC users would need to be involved in the definition and development stages of the program. Many measurement attempts of others have failed for lack of keeping that and the following in mind. Based on research by NAWC personnel and previous experience, NAWC decided any measures that were to be included in its set of measures should satisfy the following:

- Management issues should drive the measures and data collected. To avoid collecting unconnected or non useful data, thereby unnecessarily burdening the development process, the measures and data collected should be tightly coupled to issues and problems that project management needs additional insight into to make better decisions.
- The measures should be objective and repeatable. The measures should not be subjective data, but instead, should be data that represent a quantification of the process being measured. The data should be thoroughly well defined so that what it represents and how it will be collected are clear and unambiguous. The measures should not be 'invented' or 'created' as mathematical objects, and thus removed from the actual software process. Through the definition of the data, the data collection process should be repeatable so that when two or more people collect the same data, the results will be the same.
- The indicators derived from the measures should imply goodness. For measures of goodness, higher numbers should indicate a better process; and for measures of errors/discrepancies, the lower the number the better the process.
- The measures should suggest corrective action when corrective action is needed. The purpose of a measure is to improve the software development process. Accordingly, the connection between the measures and the corrective action should not be open to broad interpretation.
- The measures should be simple. With complex measures, there are often many interactions and interdependencies, thus confusing the measures' meaning and making it relatively easy for someone to misuse the measures. With a complicated measure, it may be difficult to capture the underlying process.
- The measurement data should be both measurable and predictable. The measurement data must be an artifact of the process and meaningful in terms of the underlying process. And, measurement data must be predictable for management to use it; just as schedule and budget are estimated and controlled, so should these measures. Later measurement would then improve both planning and accuracy and suggest corrective actions.

2. Development Strategy for the Measurement Process

NAWC by working in cross-divisional teams, and with guidance from the SEI, established a software measurement process action team (SMPAT) to develop the software measurement process and methods for NAWC. The SMPAT began by developing a charter that described its goals, membership, meeting logistics, and expected products. The charter was followed by generation of the SMPAT action plan, which further defined the goals and described the process by which the SMPAT would pursue those goals.

2.1 Software Measurement Process Action Team

The primary initial resources for accomplishing work on the software measurement project were the SMPAT chair and the Software Engineering Institute representative. Additional SMPAT members contributed to the extent possible. The SMPAT worked collaboratively with members of the NAWC metrics office; all metrics office staff were included as members on the SMPAT. Within this arrangement, metrics office staff members completed the primary work items and metrics office members of the SMPAT provided an important service by reviewing and commenting on software measurement products.

The SMPAT met for approximately two hours every two weeks. The purpose of the meetings was to discuss status and issues. Meeting minutes were documented and distributed among the members. Action items were noted and the status reviewed at each SMPAT meeting. The meetings became an important key to the success of the measurement initiative at NAWC by keeping the SMPAT members focused on software measurement and involved in its development process.

The SMPAT linked itself to other NAWC efforts as shown in Figure 2-1. The link to NAWC management provided the strategic direction and resources to move NAWC toward a higher maturity level through process improvements. A key to process improvement was being able to measure the process and its products in order to identify areas for improvement. Management's role was to help guide and direct the measurement and definition efforts. The SMPAT worked with the process definition effort to develop the "hooks" in the software process for software measurement.

The SMPAT leveraged NAWC projects in three ways:

1. Explored what measurements the projects were already collecting and how the projects were using measurement. The SMPAT used this information to help guide its work and recommendations while working collaboratively with the metrics office to develop measurement products.
2. Pilot tested its measurements and process on projects.
3. Provided guidance on implementing software measurement and support to projects.

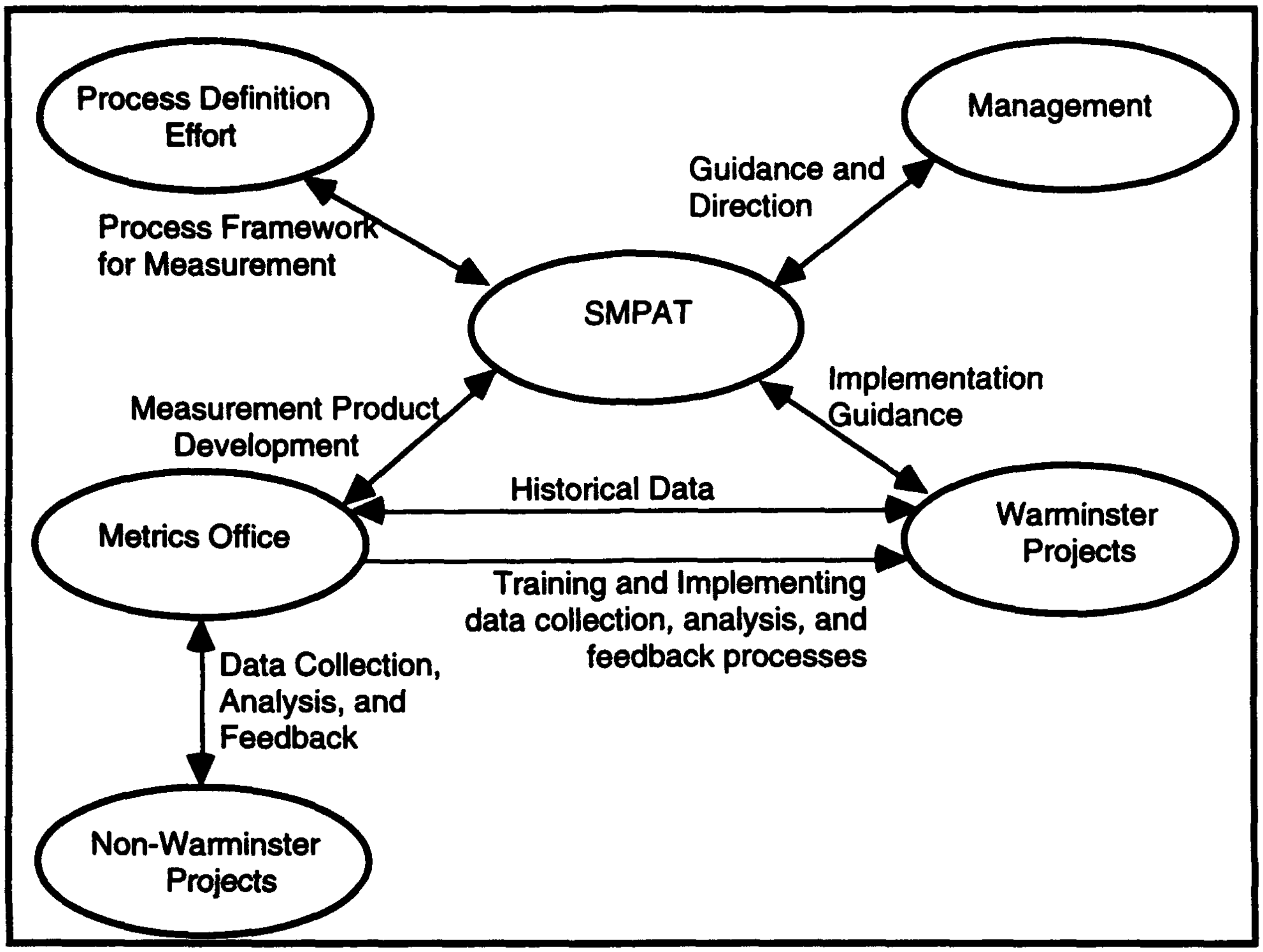

Process Definition Effort
Management
Guidance and Direction
Process Framework for Measurement
SMPAT
Implementation Guidance
Measurement Product Development
Historical Data
Metrics Office
Warminster Projects
Training and Implementing data collection, analysis, and feedback processes
Data Collection, Analysis, and Feedback
Non-Warminster Projects

Figure 2-1 Linkage of SMPAT to Other NAWC Activities

2.2 SMPAT Action Plan and Implementation Strategy

The SMPAT developed an approach that used an adaptation of the goal-question-metric paradigm [BASILI] to determine the end data items to be collected. So that the SMPAT did not proceed haphazardly, it developed an action plan that included a six-phase approach towards completing its work. (The six phases are discussed later in this chapter.) The six phases were:

1. Research and planning.
2. Establishing key management concerns.
3. Defining measurement data.
4. Defining data collection and analysis process.
5. Pilot testing measurement process.
6. Conducting software measurement training and giving awareness briefings.

After completing phases 1 and 2, the SMPAT divided its work into four increments which are described in Section 2.4. The increments partitioned the SMPAT's work logically by management concern. Each increment was to be developed using phases four through six. These four increments were described in a short plan that accompanied the action plan.

The SMPAT then began working to the activities in its plan. Its first major technical product was the NAWC software measurement guide, dubbed SR-1 [ROZUM 92b]. The purpose of SR-1 was to define the first increment of software measures, describe a detailed data collection process, and finally, describe the analysis that would be performed to make the data useful to the projects.

The SMPAT wanted to evolve its products. It incorporated evolution into its process by initially concentrating on a few key management issues, testing its recommended measurements and process, then expanding by improving and elaborating existing products and developing new ones. The SMPAT strategy consisted of the six phases identified earlier and described in detail below.

2.2.1 Phase 1: Exploration and Planning

The primary objectives of the exploration and planning phase were (1) to plan the project and its remaining phases, and (2) to determine the state of the practice in software measurement in Department of Defense (DoD) organizations and the general software community. This phase was necessary to use SMPAT members' time efficiently by avoiding duplication of previous and existing efforts and by using the experience of others as leverage.

The primary product of this phase was the SMPAT's action plan. The action plan was the vehicle for documenting the plan of the remaining activities for the measurement initiative. The exploration and planning phase was initially used to explore other similar measurement efforts; however, to improve NAWC's software measurement process continually, the project planned to explore new methods and measurements as they became visible or when a need was identified.

The SEI worked jointly with SMPAT members to perform a high-level assessment of current measurement work (both at NAWC and throughout the industry) to prepare the measurement project. A number of Warminster projects, other navy measurement efforts, as well as other DoD related efforts were explored.

During this phase, the SMPAT developed its charter and action plan. In addition, it developed other informal deliverables including:

- Report on other DoD measurement activities explored.
- Project library of relevant materials received.
- Project meeting minutes.
- Action item log.
- Members' trip reports on related topics.

This phase started in August, 1991 when the SMPAT held its kickoff meeting. Although activity in this phase never officially ceases, and hence, the phase never ends, the level of activity needed to explore other measurement efforts was reduced when the data definition phase (phase 3) started.

2.2.2 Phase 2: Establishing Key Management Concerns

To be successful, the measurements needed to be driven by the concerns and issues of NAWC management. The intent of this phase was to produce a list of prioritized management concerns.

The SMPAT drafted a strawman list of concerns and presented the list to management. Through discussions, the list of concerns and issues was modified then prioritized by the SMPAT and reviewed with NAWC Code 70 management. The SMPAT, using this prioritized list, planned the scope of its remaining work in terms of the four incremental builds.

2.2.3 Phase 3: Defining Measurement Data

The objective of this phase was to identify and define, in detail, software measures that would provide insight for each management concern identified in the previous phase. During this phase the SMPAT developed a detailed definition of each software measure and traced that measure to a management concern.

The goal of this phase was to have issues drive the software data that are collected. The SMPAT first established what the key issues and questions were for NAWC software managers. After determining the list of key management issues and questions, the SMPAT defined software measures and data parameters to collect to help address those issues, recognizing this was only an initial set of measures because not all issues or questions can be predetermined. The SMPAT then used those definitions as the basis for determining how to collect each of the data parameters.

This phase started with the completion of the action plan developed in phase 2. The phase concluded when all identified management concerns of a particular incremental build had software measures that were traceable to those concerns identified and defined.

2.2.4 Phase 4: Defining Data Collection and Analysis Processes

The objective of this phase was to define the processes to collect and analyze the data. After the definition of the data to be collected, the SMPAT felt it was important to define the process of collecting the data. The data collection process included:

- How the data would be collected (e.g., data collection forms, automated tools).
- Where the data would come from (e.g., contract data requirements list (CDRL) item for data, other CDRL items, files, reviews).
- Who would collect the data (e.g., SMPAT, project, contractor, IV&V).
- When the data would be collected (e.g., monthly, weekly, periodically).

The collection process was considered very important because if the defined data could not be feasibly collected, the data definition was useless. Likewise, the SMPAT put an equal importance on the process of how the data would be analyzed, because, if the data could not be used to help program managers make decisions, it would be principally useless. Examples of what was defined for the analysis process include: what indicators would need to be generated to address program manager concerns and issues, what types of trends could be determined, and what types of insight were needed by management to make decisions.

This phase started with the completion of the software measurement definitions developed in the previous phase and concluded when the processes for data collection and analysis were defined.

During this phase, the SMPAT developed a guidebook that traced the management concerns to the data collected, and then to the analysis of the data that addressed management concerns. Specifically, the guidebook documented:

- Management concerns.
- Data definitions.
- Data collection process.
- Data analysis process.
- Analysis results/feedback process.

2.2.5 Phase 5: Pilot Testing the Measurement Process

Before final publication and release of its software measurement work, the SMPAT desired to test its measures and processes on active NAWC projects. This would establish that the measures and processes (1) addressed the issues and concerns of management, and (2) worked within the NAWC environment.

The SMPAT identified three potential NAWC projects to apply the software measures and processes. The SMPAT chose one of the projects and then worked closely with that project and its contractors to tailor its processes to the processes already in use. The intent was to implement the measures on the project, observe the results, and modify the measures or measurement process as necessary. Chapter 4 discusses in more detail the pilot test and results of this phase.

This phase started with the identification and selection of the project. During this phase, the SMPAT developed pilot test plans and a data collection guide specific to the project. This phase ended when NAWC began to accept the software measurement application role for Navy projects on a fee-for-service basis.

2.2.6 Phase 6: Conducting Software Measurement Training and Awareness Briefings

After being satisfied that the process for the set of measures in an increment was relatively stable and could be implemented throughout NAWC, the NAWC metrics office began to develop materials for:

- Promoting awareness and understanding of its work.
- Encouraging other NAWC organizations to use software measurement.
- Training projects and individuals at NAWC to use the measurements and processes defined by the SMPAT.

The products developed during this phase had two objectives: (1) to obtain and maintain sponsorship of the SMPAT and the metrics office, and (2) to develop the materials for training and solicitation of software measurement work from projects at Warminster and eventually other Navy or DoD organizations.

This phase was planned to be continuous after pilot testing the processes and measures of the first incremental build.

During this phase the SMPAT began developing:

- Status briefings.
- Various awareness briefings.
- Marketing briefings.
- Training materials.

2.3 Identify Management Issues and Concerns

During phase 2 (establish key management concerns), the SMPAT and NAWC management worked together to assemble a list of management concerns and issues for the NAVAIR community. NAVAIR and NAWC managers had an existing task of surveying the management of the NAVAIR community to establish their concerns. The results of this survey, along with other similar surveys, and SEI knowledge of persistent problems in developing software, contributed to the initial list of issues discussed between the SMPAT and NAWC management. That list was narrowed down and prioritized by the SMPAT.

As with the development of other system components, the key issues identified for software development related to resource adequacy and expenditure, development progress and schedule, and the quality of the interim and final products. Unlike the development of other system components, however, the nature of software development causes these three areas to be strongly related. Issues with software quality, for example, can in many cases be directly attributable to schedule and resource constraints. This interrelationship between process, schedule, and quality required that the software measurement process address the relationships between seemingly disparate types of software data.

2.3.1 Resource Issues

The resource issues revolved around the adequacy and expenditure of the resources. The SMPAT intended that the following issues and concerns would be addressed by the software measures. The items in bold italics were those addressed at least partially by the measurement process developed in increment one:

- ***Staff availability and stability***
- Funding adequacy
- ***Spending rate***
- ***Allocation of resources to key activities (e.g., documentation, configuration management, or testing)***
- Productivity rate assumptions
- ***Size estimate accuracy***
- Subcontractor allocations
- Computer resource adequacy

2.3.2 Progress Issues

The progress issues are intended to address a common concern of all software managers: Will the contract be completed on schedule, and if not, when will it be completed? Questions about the progress of software contracts that NAWC management typically asked included:

- ***Schedule (milestone commitments); is the contractor meeting commitments on time and within a prescribed level of quality?***
- ***Development progress; what is the true (objective) progress of the contract?***
- ***Schedule and forecasts to completion; were they realistic?***
- ***Functionality allocations; were they shifting from earlier to later builds?***
- Productivity rate; was the planned productivity rate being achieved?
- Rework; were high levels of rework having an impact on progress?

2.3.3 Quality Issues

The technical quality issues included both the quality of products being produced and the quality of the processes used to produce the products. The managers were concerned that interim products be stable and complete so that successive processes using those products did not compound mistakes (i.e., defects) and thereby drive up the cost through added rework. Questions about the technical quality that NAWC managers have asked revolve around:

- Utilization of target computer resources (e.g., spare capacities and throughput).
- ***Problem reports (e.g., severity of problems, number of problems, timely resolution of problems, or high density levels of problems in products or processes).***
- Completeness (e.g., product and process exit criteria).
- ***Product stability and volatility (e.g., requirements, design, and functionality allocations).***
- Process stability and volatility (e.g., procedures and standards conformance).
- Premature release of products (e.g., adequately testing the software before being released).
- ***Rework (e.g., later processes finding defects caused by earlier processes).***

2.3.4 Pre-Contract Award Issues

Before awarding a development contract, NAWC managers also have raised many concerns and questions about the following:

- Is a particular bidder's proposal realistic?
  - Did the most technically capable bidder get the contract?
  - What was the technical scoring range?
  - What was the cost scoring range?
  - How did a bidder's scores compare to the expected or minimum scores required?
- Is the bidder capable of doing the job?
  - How much risk is involved and what are the high risk areas?
  - Was a software capability evaluation (SCE) used to identify risks?
  - What risks did the SCE identify?
  - Is there a management or mitigation plan for the risks?
- Are the software estimates credible?
  - Were the estimates based on the contractor's historical data?
  - Were the estimates based on industry "standards"?
  - What were the underlying assumptions of the estimates?
  - How does the program office's estimate compare to the contractor's?
  - How does the government's independent estimate compare to the contractor's?

- Is the program plan and technology mature enough to launch?
  - Is the system feasible?
  - Has the system concept been proven?
  - Is there a similar precedent system?
  - Are the system requirements defined?
  - Are the system requirements properly allocated to hardware and software?
- Do the requirements satisfy the customer's/end-user's expectation?
  - Has the customer/end-user signed off on requirements?
  - How many questions or action items did the customer/end-user have?
  - Are the customer/end-users involved with the source selection evaluation board (SSEB)?

As stated earlier, the software measures in increment one did not address each and every issue identified above. Additionally, the SMPAT conceded that this list of issues would not be exhaustive. The above list would provide only the "target" areas on which the SMPAT would focus its initial, and future, measurement development increments. Also, as the NAWC metrics office gained experience analyzing data regarding identified issues, other, perhaps more specific or critical, issues would become apparent, and additional data items and analysis would be needed.

2.4 Description of Development Increments

So that the measurement process would be effective, the software measures chosen were based on NAWC management issues and concerns. The data collected for the measures would be used to establish trends and provide information to NAWC management so that it could make decisions based on information about how the products and processes were progressing, both qualitatively and quantitatively.

The SMPAT identified four increments for implementing the measures. The SMPAT partitioned the increments logically by management issues. The first two increments were to address management issues concerning the management and control of software intensive contracts.

The first of these increments used measures described by the SEI in its technical report, *Software Measurement Concepts for Acquisition Program Managers* [ROZUM92a]. The first of these four increments was completed, and an SEI special report was written that documented the data to be collected to support the measures, the analysis methods to be used to analyze these measures, and the management issues and concerns those measures address in the first increment [ROZUM92b]. The four increments and the general measures that were or will be defined and implemented for each increment are:

Increment 1 (Management - Tier 1):

Staffing

Effort

Work progress

Product delivery (milestones)

Size

Post deployment work

Increment 2 (Management - Tier 2):

Complexity

Computer resource utilization

Requirements and design stability

Supportability

Increment 3 (Pre-award/Launch Readiness):

Requirements feasibility

Estimate realism

Launch realism

Increment 4 (Development Process Improvement):

SEI capability maturity model (CMM) activities

The second increment of measures would also address management issues for managing and controlling of software intensive contracts; however, the measures in the second increment are more complex and not easily defined or collected. Therefore, these measures were deferred to the second increment so that progress could be made on less complex processes and methods. The third increment was to address NAWC management issues that relate to pre-contract award or source selection. The fourth increment was to address NAWC's desire to move toward higher maturity levels as described by the SEI's Capability Maturity Model (CMM) [PAULK].

This incremental approach had three advantages: (1) NAWC could start to define and collect data to pilot test measures more quickly by starting these measurement processes as soon as the measure and a collection process for obtaining data was defined, (2) the approach leveraged some known methods and did not require NAWC or the SEI to research extensively or develop new measures, and (3) the increments defined a path that projects and/or organizations not currently using software measurement could logically follow and mature their measurement processes.

The SMPAT chose the six measures in increment one because knowledge was readily available about each and the issues addressed by the measures were key NAWC management concerns. Each of the measures in increment one is discussed in detail in Chapter 3 of this report. Increments two to four were not developed at the time of this report, but are described conceptually below.

2.4.1 Description of Increment Two Measures

The measures and data to be collected for this increment are intended to further support management issues regarding the management of a software development or maintenance project.

The measures in increment two were included here rather than increment one for predominantly one or both of the following two reasons:

1. The SMPAT felt that certain measures were straightforward and easy to develop. This would allow the SMPAT to develop the processes to support a measure quickly and show some successes early. Because of this, some measures were moved into increment two because the SMPAT did not believe they could easily define the measure or a method to collect the data necessary to support it.
2. The insight that an increment two measure provided was a lower priority than some of the others. An example of an increment two measure that follows the second line of reasoning is computer resource utilization.

The measures to be defined and collected for increment two then include:

- Complexity
- Computer resource utilization
- Requirements stability
- Design stability
- Supportability

Complexity

The complexity of the software would give the program manager additional information that could be used with the increment one measures to provide more insight into the amount of work that needed to be completed and the cost of that work. For example, with increment one measures, the manager has insight into the size, effort, and amount of work that needs to be completed. But by adding information on complexity, the manager would also have information on why some software modules of comparable size to others might take longer or be expected to have more errors.

Computer resource utilization

The computer resource utilization is intended to give the program manager early insight into whether or not the requirements for computer resources will be met. For example, the software may have performance requirements or may need to allow for future expansion all within a known hardware configuration.

Requirements and design stability

The requirements and design stability measures are intended to give the program manager insight into whether or not the products of the requirements analysis and design process are stable. The stability of these products will directly affect the overall schedule and cost of the project if later products have to be reworked because of changes in the requirement or design products (e.g., a design or requirement specification).

Supportability

The supportability measure is intended to give the program manager information about the requirements (e.g., testing, cost, and schedule) to support a specific software product. The intent of the SMPAT was to leverage work completed by the United States Army Material Command's Missile Command [MOORE]. That work has developed an assessment model that provides an index describing the supportability of a software system.

2.4.2 Description of Increment Three Measures

The measures and data to be collected for increment three are intended to support management during the pre-source selection phase, i.e., prior to the release of a request for proposal (RFP). The measures in this increment would predominantly require research and possibly the development of new measures, methods, and processes never performed before. Each of the measures below was envisioned as an index with implications of the risk involved in developing a proposed system. The measures for increment 3 include:

- Requirements feasibility
- Estimate realism
- Launch realism

Requirements feasibility

The requirements feasibility measure is intended to address issues regarding the precedence of the software or functionality for the system to be developed. The intent is to determine quantitatively whether or not the software requirements are "doable" and/or whether a risk exists in assigning certain functionality to the software of the system. The idea is for NAWC to ensure that software functionality requirements stated in an RFP are, in fact, feasible and that software can be developed to meet its needs.

Estimate realism

The estimate realism measure is intended to address issues regarding risks in cost and schedule overruns of systems. Of the three measures in increment three, this measure would require the least research, but instead, require the buildup of historical data from increments one and two from which comparisons and evaluations could be made with a proposed system.

Launch realism

The launch realism measure is intended to address issues regarding the capability to proceed with the a new system. The launch realism measures were envisioned as an index and are based on:

- Information about the capability of the technology required and available to develop the system.
- Capability of the government program office including the funds to build the system adequately.
- Capability of the contractor to develop the system based on its history of developing similar systems, or complex and/or unprecedented systems.

2.4.3 Description of Increment Four Measures

The measures to be collected and analyzed for increment four would be based on the Capability Maturity Model (CMM) developed by the SEI [PAULK]. The measures would leverage work already completed by the SEI and support NAWC's efforts to improve its software process.

3. Description of Increment One Measurement Processes

The first section in this chapter identifies the issues addressed by the measures in increment one. The following sections describe how the data were defined, collected, and analyzed, and how the results of data analysis were verified and then reported to the government program manager. Note that, unlike typical measurement efforts that involve a government agency and the contractor community, NAWC collected as much of the data as it could on its own, i.e., without contractors delivering specific "metrics" data. The following were data collected for each project participating: staff-hour data, dates of milestone reviews and product deliveries, work progress data, problem report and change proposal data, and size data.

3.1 Issues to Address

As discussed earlier, the issues in increment one were related to the government's management and oversight of software projects. The issues were further partitioned into resource issues, progress issues, and technical quality issues. The method(s) for analysis to address each issue are described in the following sections. The specific issues that were addressed (albeit to varying degrees) by the measures in increment one included:

- Resource issues
  - Staff availability and stability.
  - Spending rate.
  - Allocation of resources to key activities (e.g., documentation, configuration management, or testing).
  - Size estimate accuracy.
- Progress issues
  - Schedule (milestone commitments); was the contractor meeting commitments on time and within a prescribed level of quality?
  - Development progress; what was the true (objective) progress of the contract?
  - Schedule and forecasts to completion; were they realistic?
  - Functionality allocations; were they shifting from earlier to later builds?
- Technical quality issues
  - Problem reports (e.g., severity of problems, number of problems, timely resolution of problems, or high-density levels of problems in products or processes).
  - Product stability and volatility (e.g., requirements, design, and functionality allocations).
  - Rework (e.g., later processes finding defects caused by earlier processes).

3.2 Data Definitions

The SMPAT developed data definitions for the measures in increment one that described the data in unambiguous terms. Each measurement definition would describe why a program manager might want the data and how the measure supported specific management issues.

The definitions that follow were those preferred by the NAWC metrics office. However, the metrics office would accept (at contract award time) a definition that was different or tailored for a particular contract. The only constraint was that the data must have the capability to be partitioned similar to that described in the preferred definition. For example, the metrics office would accept a source line of code (SLOC) definition other than that defined in data definition form number 3 as long as the SLOC data were partitioned by language and also by whether it was new, modified, or reused. Data definition form 3 is included in Appendix B.

Staffing

The staffing measure provides the manger with insight into the staffing labor categories used on the contract. This information was used to track the contractor's ability to maintain a sufficient level of staffing to complete the contract [BOEHM] and to track the amount of the contractor's staff turnover.

The contractor's full-time and part-time staff was tracked according to defined and agreed-upon labor categories. Also tracked were the losses (staff turnover) and additions by labor category. Losses that resulted in a higher than expected staff turnover for the project could have jeopardized the project's quality and expected productivity rates because replacement staff would need to be trained to become familiar with the software being developed and with the decisions that had been made regarding that software.

To be counted as a full-time staff member, a person would have to expend more than 80 staff hours on the contract within a single labor category in a calendar month. Each contractor staff member who left the contract (e.g., the person is assigned to another contract or leaves the contractor's organization or company) or a contractor staff member who did not expend at least 10 staff hours on the contract in the current month and in the previous calendar month (i.e., 10 hours in each month), was counted as a staff loss. Staff members who expended effort on the contract, but could not be counted as full-time staff nor as losses, were counted as part-time staff members.

The staffing measure also provided a manager with insight into the number of full-time and part-time people working on the contract. When used along with the effort measure, the staffing measure would indicate an approximate amount of overtime being worked (regardless of whether or not it is compensated). The actual staff levels of the contractor's labor categories were tracked and compared to the levels that were planned to be used.

The experience profiles of the labor categories for potential contractors to address were documented in the request for proposal (RFP). To help define the labor categories, data definition form 1 would be used. Typically, a minimum of four labor categories were defined using data definition form 1 and were checked for consistency with the prime contractor's statement of work (SOW). Data definition form 1 is included in Appendix B.

Alternatively, the contractor could track staff levels by labor category using a work breakdown structure (WBS). If a WBS was used, it would have to be used for the effort measure also. Using this method, data would be collected for WBS elements at the level that software activities were defined.

Effort

The effort measure tracked the staff hours being expended by the contractor during the various development activities and for the entire project. Since software development is a human-intensive and -dependent activity, the effort expended was typically the largest and least controllable cost variable. The effort measure showed the relationship between planned versus actual staff hours expended and was used to monitor whether work products were being developed according to planned expenditures of resources. The effort measure was particularly useful because it alerted managers to changes to and deviations from the plan that could drive the cost up and cause the budget to be exceeded.

Each month, a contractor would report the actual staff hours expended for each labor category and by functional area (software quality assurance, configuration management, analysis, programming, etc.), using its in-house cost accounting system.

To further define the staff-hour data to be reported by contractors and subcontractors, data definition form 2 was used.[1] Data definition form 2 is included in Appendix B.

Product Delivery (Milestones)

The product delivery measure gave management a comparison of actual milestone completions against established milestone commitments. This measure quantified the contractor's performance toward meeting commitments for delivering products and completing milestones. If individual schedule commitments were met, the whole project was more likely to stay on schedule.

The product delivery measure helped to portray graphically planned delivery dates, revised delivery dates, and the intermediate activities needed to meet the end delivery dates. Ideally, each milestone would have at least one key interim activity per month. The product delivery measure could also be used to track interim events defined in the software development WBS leading up to the milestones, e.g., tool developments and government furnished equipment (GFE) or information (GFI). These intermediate activities and events were not standardized data items, but were project-specific items that were key deliverables for the project to meet its commitments.

The product delivery measure tracked the completion of the following DOD-STD-2167A milestone events when applicable on a contract:

---

[1] This form is extracted from the SEI technical report, "Software Effort and Schedule Measurement: A Framework for Counting Staff Hours and Reporting Schedule Information," CMU/SEI-92-TR-21 [GOETHERT]. See the technical report for further details and descriptions of the attributes on the form.

- Software specification review (SSR).
- Preliminary design review (PDR).
- Critical design review (CDR).
- Test readiness review (TRR).
- Formal qualification testing (FQT).
- Functional configuration audit (FCA).
- Physical configuration audit (PCA).

The delivery of the following software products were also tracked:

- Software requirements specification (SRS).
- Interface requirements specification (IRS).
- Software design document (SDD).
- Interface design document (IDD).
- Software test plan (STP).
- Software test description (STD).
- Software test report (STR).
- Software development plan (SDP).

The exit criteria for completing milestones were that the review was held and all significant action items were closed from that phase. Exit criteria for the completion of the product was the delivery of the product. Objective exit criteria for each intermediate event and activity are defined and agreed to by the manager and the contractor, typically at contract award on a project-by-project basis. Note that a level of quality was not part of the exit criteria for a product. The quality of the product, or lack of it, was tracked through the post deployment work measure. Analysis methods discussed in later sections describe how the measures were correlated to determine impacts on the project and address management concerns. In this case, if a product was delivered on time, but the level of quality was low, (e.g., there were a large number of missing paragraphs or descriptions at too high of a level of detail), the impact on the project would be determined through analysis of the product delivery measure in conjunction with the post deployment work measure—and possibly others.

For each revision to the plans, associated schedule delays were calculated using the latest approved plan as a basis. These delays were then tracked to determine the impact on the project. The delays were with respect to the original plan and all approved plans superseding the original.

Work Progress

The work progress measure gave management a quantitative indication of the progress toward completing activities. The measure used data on the planned and actual progress of work items within software development activities. The data were used to assess whether an activity was complete and the contractor was ready to proceed to successive activities.

The work progress measure tracked the progress of the development by quantifying and tracking work items completed. For each activity, counts were made of work items that have met defined exit criteria. This data, along with information on how many work items needed to be completed for that activity, provided a quantitative method to determine how much work remained. The work progress measure was applied across the major software development process activities, i.e., requirements analysis, preliminary and detailed design, code and unit test, integration and test, and formal test.[2]

As a minimum requirement, the following planned and actual data inputs were reported monthly for each activity. Ideally, the manager and contractor defined additional work items for each activity with objective exit criteria to better track work progress. Each of these following inputs would be translated into entities that reflected the contractor's process:

- Requirements analysis.
  - Number of software requirements documented in the SRS.
  - Number of software requirements documented in the IRS.
- Preliminary design.
  - Number of SRS and IRS requirements documented in the SDD.
  - Number of SRS and IRS requirements documented in the IDD.
- Detailed design.
  - Number of computer software units (CSUs) designed.
- CSU code and unit testing.
  - Number of CSUs coded and unit tested.
- Computer software component (CSC) integration and testing.
  - Number of CSC integration tests completed.
  - Number of CSC regression tests completed.
  - Number of CSUs integrated.
- Formal qualification tests.
  - Number of FQT tests completed.
  - Number of software requirements tested.

[2] If a development model such as an incremental build or spiral model was used to develop the software, then each increment would be tracked separately.

The exit criteria for each of the above work items are described in Table 3-1 below. Note that most of the exit criteria include a criterion for no open action items. In this technical report, action items also includes formal tracking items such as software change proposals and software problem reports as well as action items from reviews, walkthroughs, and inspections.

Activity	Work Items	Work Item Exit Criteria
Requirements analysis	Software requirement documented in SRS	1) Review held (e.g., walkthrough) and reviewers agree requirement adequately addressed in SRS. 2) No open action items that would impact that requirement.
	Software requirement documented in IRS	1) Review held (e.g., walkthrough) and reviewers agree requirement adequately addressed in IRS. 2) No open action items that would impact that requirement.
Preliminary design	SRS requirements documented in SDD	1) Review held (e.g., walkthrough) and reviewers agree requirement adequately addressed in SDD. Implies that requirement is traceable from SRS to SDD. 2) No open action items that would impact that requirement.
	IRS requirements documented in IDD	1) Review held (e.g., walkthrough) and reviewers agree requirement adequately addressed in IDD. Implies that requirement is traceable from IRS to IDD. 2) No open action items that would impact that requirement.
Detailed Design	CSUs designed	1) Design walkthrough complete. 2) No open action items that would impact that CSU.
CSU code and unit testing	CSUs coded and tested	1) CSU compiled without errors. 2) Unit test completed without errors. 3) Code inspection completed that included a review of unit test results. 4) No open action items that would impact that CSU.

Table 3-1 Exit Criteria for Software Work Items

Activity	Work Items	Work Item Exit Criteria
CSC integration and testing	CSC integration tests	1) Successful completion of all test steps in the software test description for that CSC. 2) Review of test results completed. 3) No open action items that would impact that CSC or its CSUs.
	CSC regression tests	Successful completion of identified tests.
	CSUs integrated	Successful completion of the CSC integration test.
Formal qualification testing	FQT tests	1) Successful completion of all test steps in the software test description for that FQT test. 2) Review of test results completed. 3) No open action items that would impact that CSCI, or its CSCs or CSUs.
	software requirements tested	1) Requirement traced from SRS or IRS to a test step in the software test description and that test step successfully completed. 2) No open action items that would impact that software requirement.

Table 3-1 Exit Criteria for Software Work Items (continued)

Size

Software size data gave the program manager an indication of the size of the software being developed. Software size was used as an indication of the amount of work to be done and the amount of resources needed to do the work. The size measure would alert the program manager to changes in the estimate(s) and/or actual software size. As the size of the software grew beyond that which was expected, the schedule and, therefore, the budget would likely be exceeded.

The actual software size was tracked and compared to the estimate(s), then analyzed for trends that the size of the software was growing or that functionality was moving from earlier to later builds. Estimates of various size attributes were compared with new estimates or actual code size monthly throughout the life cycle of the program.

The size data consisted of the number of CSUs and source lines of code (SLOC). SLOC was counted for each software language used during development. Each language was further partitioned by new, modified, and reused SLOC.

NAWC defined SLOC (and what was included in SLOC) as follows:[3]

> *Source lines of code (SLOC):* The lines of code that were generated by the software developers and that can be classified as statements. Each statement separated by a language-specific delimiter was counted as one SLOC, regardless of whether the statement was contained on one line or continues across multiple lines. A single SLOC statement may contain one or more embedded comments and still be counted as one SLOC. Statements include: declarations, executable statements, format statements, compiler directives, constructs, and data lines.

NAWC defined declarations, executable statements, format statements, compiler directives, constructs, and data lines as follows:

> *Declarations:* A declaration associates an identifier with an entity. Examples include static initializations, type and bounds declaration, variable definitions, declarations of constants, procedure headers and argument lists, function declarations, interface specifications, import statements, and statements that declare or use generics.
>
> *Executable statements:* Executable statements produce run-time actions or control program flow. They contain the operational logic of the program. One characteristic of executable statements is that they can be stepped through with interactive debuggers. Also, debuggers can set breakpoints with respect to executable statements.
>
> *Format statements:* Format statements are statements that provide (often static) formatting rules and textual labeling for information that was to be displayed on screens, printed on output reports, or sent to files or other output devices. They are also used to define the ways in which input information will be interpreted and edited.
>
> *Compiler directives:* Compiler directives are special instructions to compilers, cross-compilers, preprocessors, linkers, translators, etc.
>
> *Constructs:* Constructs are those statements that provide some sort of control structure. Constructs are usually reserved words within a language and include statements such as: if-then-else, do-continue, goto, call(), with, procedure, function, case, etc. Constructs such as begin-end, for-next, etc., are counted as one SLOC.
>
> *Data lines:* Data lines are either (1) those statements that are used to specify an initial value for variables, to fill an array with data elements, or (2) the data items of a single database record. Each data line separated by a delimiter was considered one SLOC.

[3] The description for declarations, executable statements, format statements, and compiler directives was adapted from "Software Size Measurement: a Framework for Counting Source Statements," CMU/SEI-92-TR-20 [PARK]. The description of constructs and data lines is adapted from *Code Counting Program Requirements Document,* by Keystone Computer Associates, Inc. prepared for NAWC, August 6, 1991.

To further define SLOC reported by contractors and subcontractors, data definition form 3 was used to define what was included in the SLOC data.[4] Data definition form 3 is included in Appendix B.

The following guidelines were adapted or extracted from [JONES] and are specific examples of how certain code syntax would be counted:

- Do not count commentary lines or statements that were used purely for information purposes; i.e., the following statement would not be counted:

  ```
  REM THIS ROUTINE CALCULATES GROSS PAY
  ```

- For strongly typed languages, count data definitions statements as one statement; i.e., the following example counts as one statement:

  ```
  WORKDAYs = (MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY)
  ```

- For weakly typed languages, count data definitions statements as one statement; i.e., the following example counts as one statement:

  ```
  DATA = MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY,

  SATURDAY, SUNDAY
  ```

- Count assignment statements as one statement (i.e., A = B + C).
- Count equations as you would treat them in mathematics, (i.e., as a single unit or line). The following expression counts as one statement:

  ```
  A = ((B * C) / (D - E)) + ((F + G + H) * I))
  ```

  however, the following would be counted as three statements:

  ```
  A = B = C = Pi
  ```

- For languages that allow multiple logical statements per physical line, count the logical statements similar to counting delimiters. The following example counts as three statements:

  ```
  BASE = 0:  BASE = HOURS * RATE: PRINT BASE
  ```

  Note: The two colons and the carriage return at the end of the line are the three delimiters.
- Count procedure or function calls as one statement, (i.e., CALL PRINTDRV).

[4] This form is extracted from the SEI technical report, "Software Size Measurement: a Framework for Counting Source Statements," CMU/SEI-92-TR-20 [PARK]. See the technical report for further details and descriptions of the attributes on the form.

- Count IFs and THENs as one statement each. The following example would count as two statements:

```
IF A OCCURS THEN UPDATE B
```

- Count IFs, THENs, and ELSEs as one statement each. The following example would count as three statements:

```
IF A OCCURS THEN UPDATE B ELSE UPDATE C
```

- Count ELSEIF statements as one statement.
- Count CASE statements as separate statements. The following example would count as five statements:

```
SELECT CASE NUMBER$
CASE "ONE": PRINT "1"
CASE "TWO": PRINT "2"
CASE "THREE": PRINT "3"
CASE "FOUR": PRINT "4"
END SELECT
```

- Count DO-WHILE constructs as 1 statement. The following example would count as five statements:

```
WHILE J>10 DO
PRINT HEADER
PRINT LINENUM
PRINT MESSAGE
J = J + 1
LOOP
```

- Count FOR-NEXT constructs as one statement. The following example would count as three statements:

```
FOR J=1 TO 10
PRINT HEADER
PRINT MESSAGE
NEXT J
```

- Count REPEAT-UNTIL constructs as one statement. The following example would count as four statements:

```
REPEAT
PRINT HEADER
PRINT LINENUM
PRINT MESSAGE
UNTIL INDEX >10
```

- Count nested constructs in accordance with the individual element rules. The following example would count as six statements:

```
FOR J=1 TO 10
PRINT HEADER
FOR K=1 TO 3
PRINT NUMBER
NEXT K
PRINT LINENUM
PRINT MESSAGE
NEXT J
```

Post Deployment Work

The post deployment work (PDW) measure was used to track the evolving quality of the products as measured by the number of closed and open software problem reports.[5] Using this measure, the manager could determine the level of resources needed, the progress made, and the technical quality of the software and the processes used to develop the software. This measure also gave the manager an early indication of the future requirements to support the software.

The post deployment work measure also gave managers information on the readiness of the software to proceed to the next phase by tracking the acceptability and problem content of the products. The information on problem reports would be used to determine whether to move products forward to the next development activity or to continue the current activity and further mature the products. For example, if an inordinate number of SPRs were open at TRR, a recommendation may include that the contractor needs to spend more time maturing the software before continuing on to the formal qualification testing.

Software problem reports (SPRs) were discrepancies in a product noted during the development process. The intent was to initiate and track SPRs for any product or work item that had passed through its exit criteria and was counted as completed by the work progress or product delivery metric. However, since NAWC is a government organization collecting data on a contractor's process, this intent was not met in all cases. It is believed, though, that when properly inserted on contract, the intent would be met. For the PDW measure, data was collected on the total number of SPRs received, opened, and closed for the month and overall. SPR data were also collected for the following:

[5] Often, projects tracked software problem reports and software change proposals. When data on software change reports were available, NAWC would collect and report the data similarly to the software problem report data, but separately.

- For each CSCI, SPRs were reported by priority.
- For each priority level, SPRs were reported by longevity.
- For each test source that can discover software problems and open SPRs, SPRs were reported by priority.

An SPR was considered opened when it was received by the organization designated to accept it. An SPR was closed after all signatures or approvals needed to close it were obtained; the signatures and approvals minimally included the manager responsible for the correction, the contractor's SQA or testing representative, and the government person responsible for that project.

The following test sources were the process life-cycle phases where the software was when an SPR originated:

- Pre-integration testing, i.e., all activities prior to integration testing.
- Integration testing.
- Preliminary formal qualification testing.
- Formal qualification testing.
- Independent verification and validation.
- Beta site testing.
- OT&E (operational testing and evaluation).
- Field (reported by the users in the field).

NAWC used the definition of priorities for SPRs in DOD-STD-2167A [2167A].

3.3 Data Collection Process

Each month, the NAWC metrics office collected data from participating projects. That data included:

- Staff-hour data for the prime contractor and its subcontractors using data collection form 2.
- Milestone review and product delivery data using data collection form 3.
- Work progress data using data collection form 4.
- Size data using data collection form 5.
- Problem report and change proposal data using data collection form 6.

In general, the NAWC metrics office collected the data from existing processes and sources. That is, a separate CDRL item for delivering software measurement data was usually not required. If needed, The NAWC metrics office would tailor its definition to be able to collect the data in a format similar to that in data collection forms 1 to 6. The metrics office always tried to get data in electronic form and then, using various methods, parse the information to obtain the data it needed. When this approach was applied, the data collection forms may not actually be completed because the data were electronically transferred to the database.

Also, by not requiring a separate CDRL item for measurement data, the metrics office was able to collect software measurement data at little or no additional cost to the contractor. The primary cost that did exist was that needed to support the NAWC metrics office. This cost was usually paid by the government program office responsible for the system being developed. In return, that government program office received a proven software measurement process and the benefit of a continually growing experience base of software measurement practitioners.

3.3.1 Collection of Staffing Data

Monthly, NAWC received cost accounting data electronically from a contractor. These data were then read into a database and, via macros, the data were parsed into the required formats to support the staffing measure. If the data were not available electronically, the prime contractor reported its staffing data and its subcontractor's staffing data to the NAWC metrics office using data collection form 1. Data collection form 1 would be completed for the prime contractor and each of its subcontractors separately. Additionally, a separate form would be used for each build or delivery order.

Data reporting started at contract award and continued monthly for the life of the contract. From the contractor's proposal and development plans, the NAWC metrics office received the planned staffing levels for comparison with the actual data. The labor categories were defined at contract award in the SOW and were also used by the effort metric.

3.3.2 Collection of Effort Data

For the effort metric, the NAWC metrics office used macros to parse the cost accounting data received from a contractor by labor category for each functional area. The minimum functional areas for which the contractor collected data were listed on data collection form 2. If the data were not available electronically, the contractor reported its effort data and the effort data from its subcontractor(s) to the NAWC metrics office using data collection form 2. Data collection form 2 would be completed by the prime contractor and each of its subcontractors separately. Additionally, a separate form would be used for each build or delivery order.

Data reporting started at contract award and continued monthly for the life of the contract. From the contractor's proposal and development plans, the program manager received the planned staff-hour expenditure rates for comparison with the actual data. The labor categories were defined at contract award in the SOW and were also used by the staff measure.

3.3.3 Collection of Product Delivery (Milestones) Data

Monthly, the NAWC metrics office recorded actual milestone completion dates and product delivery dates using data collection form 3. Additionally, the metrics office would track interim events and products when available.

The plan data for the product delivery measure were obtained from the contract CDRL. Additional plan data were obtained from the SDP and at SSR, PDR, and CDR. When data from multiple plans were available, data from superseded plans were still kept and used during analysis to determine changes and impacts of replanning. For example, if milestone plan data were obtained from SDP and again from the SSR, both sets of data were kept. Even though the data from the SDP may be superseded by the data from the SSR, the trends from replanning would be important information needed during data analysis.

3.3.4 Collection of Work Progress Data

The NAWC metrics office collected data for the work progress measure monthly using data collection form 4. A minimum requirement was to have the planned and actual data inputs reported monthly for each build, delivery order, release, etc. The following data were collected:

Software requirements: The number of software requirements was obtained from the SRS and IRS. The number of requirements documented in the SDD would be obtained from a traceability matrix usually found in the SDD and the IDD.

CSUs: The planned number of CSUs was obtained from the SDD. The metrics office obtained data on the actual number of CSUs that had met their exit criteria for being designed and coded and unit tested from contractor status reports.

Integration tests: The planned number of integration tests for each incremental build was obtained from the STP. The number of regression tests to be executed for each build was obtained from either the STP, reported at the CDR, or recorded in the minutes of the CDR. Data were obtained on the actual number of integration and regression tests completed and the number of CSUs integrated from contractor status reports.

Formal qualification tests: The planned number of formal qualification tests and the number of software requirements to be tested was obtained from the STP. Data were obtained on the actual number of formal qualification tests completed from contractor status reports. If the actual data were not available from contractor status reports, the contractor would complete data collection form 4 each month.

3.3.5 Collection of Size Data

The NAWC metrics office collected data from contractor status reports on the number of SLOC and CSUs by CSCI for each build. If the actual data were not available from contractor status reports or software development files, the contractor would complete data collection form 5 each month.

The planned number of CSUs was obtained from the SDD. The original estimate for total SLOC was obtained from the contractor's proposal and updated formally in the SDP and at SSR, PDR, and CDR.

For each development language used within a CSCI, the number of actual SLOC under configuration control for the build of a CSCI was counted as well as an estimate of the number of SLOC remaining to complete the build for that CSCI. The actual and estimated SLOC numbers were further partitioned by the amount of new, reused, and modified SLOC. If possible, the NAWC metrics office would count the code. Often though, this was not feasible and the contractor was responsible for counting the code and supplying the data to the NAWC metrics office. The actual SLOC was usually obtained from the software development folders using an automated counting tool approved by the NAWC metrics office.

3.3.6 Collection of Post Deployment Work Data

The primary data item for the post deployment work measure was the software problem report (SPR). The SPR data were collected by the NAWC metrics office monthly using data collection form 6. Ideally, the data were obtained from an automated database management system that stored all SPR information for the project.

For the post deployment work metric, data inputs on SPRs were collected separately for the total number received, opened, and closed since the last report as well as for the total to date. Additionally, the SPR data were parsed as follows:

- Each CSCI has SPRs reported by priority (open only).
- Each test source has SPRs reported by priority (opened and closed).
- Each SPR priority level was reported by longevity (open only).
- Software change proposals (SCPs) were reported by longevity (open only).

3.4 Data Analysis Process

This section describes some suggested data analysis methods to specifically address the management issues in increment one. These methods were documented in the NAWC Software Measurement Guide [ROZUM 92b]. The methods were used to various extents on each project for which measurement was applied. The decision on what methods to use primarily depended on what issues and concerns were of priority to the program manager. That is, a project might not use all of the methods described. Regardless of the methods used, when briefed on the analysis results, projects were reminded that software measures do not answer questions, but instead, provide insight into selected areas.

Sections 3.4.1 to 3.4.3 address key analysis methods for the three core issue areas that were measured: resources, progress, and technical quality. The power of software measurement comes not as much from the analysis of a single metric—though each can be important in its own respect—but from the analysis of trends and impacts across a variety of measures and how various analyses address the issues. For more examples of using other key measurement analyses in managing programs quantitatively, see [ROZUM 92a] and [BAUMERT].

Some of the charts used for analysis include upper warning limits (UWL) and lower warning limits (LWL). The warning limits, unlike statistical control limits, were not statistically determined but were arbitrarily determined. They were simply an arbitrary percentage of planned levels. Whenever the actual data curve falls outside the warning limits, potential problems could be starting to show up and further investigation would be needed. Seemingly good trends, such as the actual data curve being higher than the UWL, were also investigated. As part of the analysis process, the NAWC metrics office investigated trends by asking probing questions, analyzing the data and other contract information more closely, and using other measures to determine the impacts of the trends on a program.

The examples shown in the following sections are *only examples.* They were not meant to imply what a good or bad project would look like. They are not meant to represent NAWC, nor were the examples taken from NAWC projects. These examples are fictional representations of data. They are meant only to be samples that illustrate how a certain chart could be constructed. Likewise, the analysis descriptions are examples of how the charts can be used as indicators; they are not a step-by-step procedure for analyzing the charts. The analysis of data was subjective; however, certain trends and indications would normally be apparent. Because the analysis was subjective, it was very important to correlate negative AND positive trends across indicators.

3.4.1 Resource Issue Analyses

The resource issues revolved around the adequacy and expenditure of the resources. Management could use the measures in increment one to gain insight into the following resource issues:

- Availability and stability of staff.
- Spending rate.
- Allocation of resources.
- Accuracy of size estimate.

Availability and Stability of Staff

To determine if the staffing levels were at planned levels and if the staff itself was stable, the metrics office analyzed data from data collection form 1 to generate charts similar to Figures 3-1 and 3-2.

To analyze the staff availability, a chart similar to Figure 3-1 would be generated for full-time staff and part-time staff for the prime contractor and each of the prime's subcontractors. The UWL and LWL in Figure 3-1 are ±25% of the planned curve.

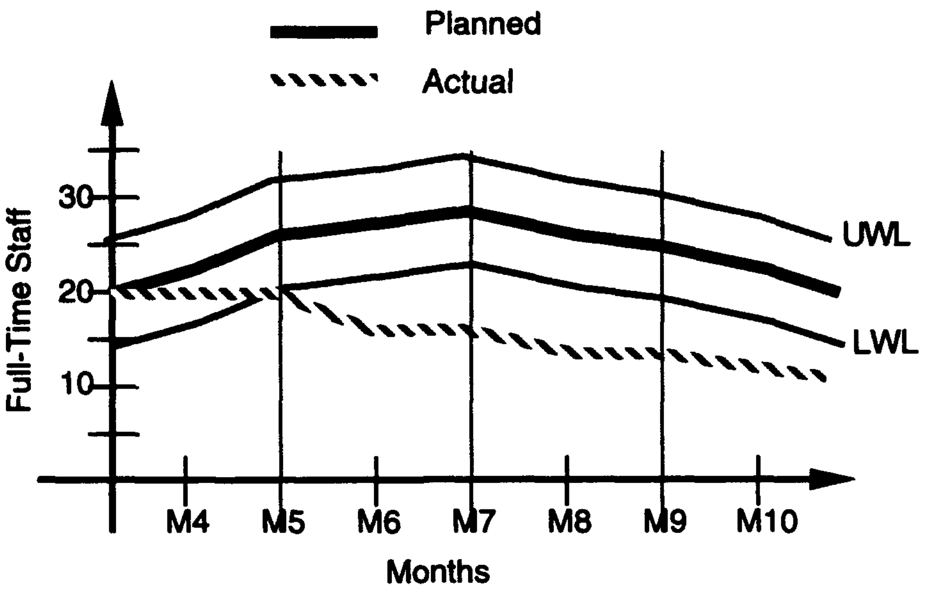


Figure 3-1 Sample Staff Availability

When actual staffing levels exceeded the UWL and LWL for either the full-time or the part-time staff charts, the NAWC metrics office would first inspect the staff chart that is not exceeding the limits to determine if the other is making up the difference. For example, if the actual curve of the full-time staffing chart is below the LWL, then is the actual curve of the part-time chart above the UWL? To address this and other questions, the metrics office analyzed the following situations:

- Case 1; full-time actual curve is less than LWL and the part-time actual curve is less than LWL. This case is potentially the worst case example. The staffing levels are not at the level which was planned to do the job. In this case, completing the project on schedule would be in jeopardy. If the project appeared to be on schedule, the quality of work being completed would be analyzed. The NAWC metrics office would analyze data from the product delivery, work progress, and post deployment work measures.

  From the work product delivery and work progress measure, the NAWC metrics office would inspect the data to determine if work is being completed as planned. The NAWC metrics office would pay particular attention to the exit criteria for the various data to ensure that milestones and work progress data items being reported were actually meeting their exit criteria. The longevity of SPRs from post deployment measure data would be analyzed to determine if rework was starting to accumulate.

- Case 2; full-time actual curve is less than LWL and the part-time actual curve is more than LWL. This case also implies that completing the project on schedule would probably be at risk. Similar to case 1 above, the NAWC metrics office would analyze data from the product delivery and work progress measure to determine if the work will be completed properly and on-time. Data from the post deployment work measure would also be analyzed to ensure that the amount of rework backlogged would not be excessive.

- Case 3; full-time actual curve is more than UWL and the part-time actual curve is less than UWL. This case would need to be analyzed carefully. If the work is being completed at a rate that is commensurate to the increased level of staffing and the technical quality level is sufficient, then the project should complete ahead of schedule. If the work is not being completed at a rate equivalent to the increased staffing level, but is being completed according to the original plan, the project may complete on time, but probably over budget. Here, the NAWC metrics office would also analyze the effort measures to determine if the spending rate had increased significantly and if the budget was in jeopardy.
- Case 4; full-time actual curve is within the UWL and LWL, but the part-time actual curve is more than the UWL. In this case, the NAWC metrics office would be concerned about what happens when the part-time actual staff levels drop back below the UWL. If the part-time staff were brought in for administrative duties to meet the schedule, were planned to be brought in and just started earlier than planned, or were just staying on, (i.e., were leaving the project later than planned), this is usually not a problem, but the NAWC metrics office should carefully monitor the budget to determine if the additional expenditures were having an excessive impact. However, if the part-time staff were added to correct technical problems, the NAWC metrics office would be concerned about what happened when the part-time staff leave the project; will others still on the project know what was done to correct problems and how to correct them in the future should they reappear?
- Case 5; full-time actual curve is within the UWL and LWL and the part-time actual curve is also within its UWL and LWL. Here, the NAWC metrics office would monitor the situation and ensure that compensating additions and deletions to the staff were not what is keeping the staffing levels stable.

To analyze the staff stability, a chart similar to Figure 3-2 was generated for the staff of the prime contractor and each of the prime's subcontractors.

The NAWC metrics office would analyze the staff turnover data from data collection form 1 using a chart similar to Figure 3-2. Primarily, the NAWC metrics office would be concerned that the contractor's staff is not changing excessively. Some changes are unavoidable and necessary; too many changes could introduce a risk of completing the project on schedule as well as a degradation in the overall quality of the products. All trends in the program's staff stability chart also would be analyzed in conjunction with the trends in the staff availability chart.

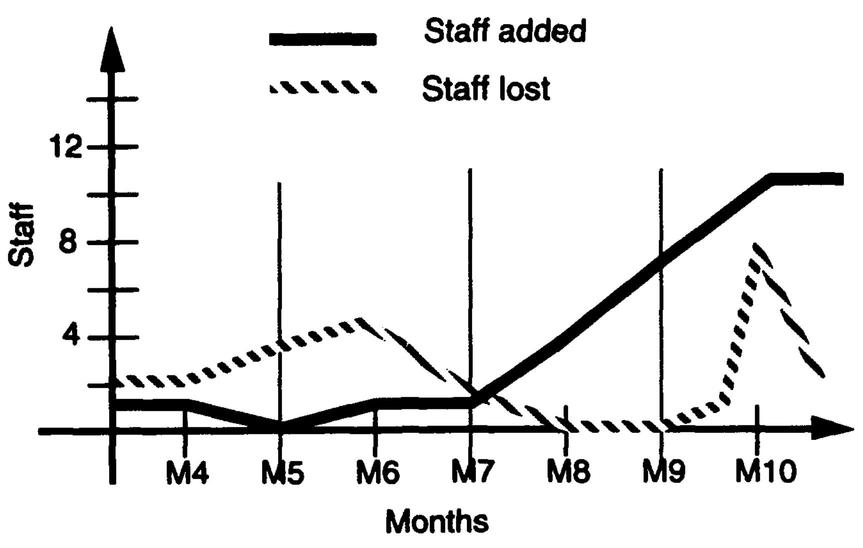


Figure 3-2 Sample Staff Stability

The trend that the NAWC metrics office would analyze carefully is when staff additions and staff losses are about the same and the actual staff availability trends are within the UWL and LWL (i.e., case 5 above). This scenario could be an early warning that the program may be facing future schedule and budget problems. Also, the maintenance of the program may be more tedious and therefore more costly and burdensome. The other various trends that the NAWC metrics office look for include:

- Large staff losses and large staff additions. This trend is similar to the trend above; however, the budget problems will occur sooner during development.
- Large staff losses and few staff additions. This is the worst case and is usually an early indicator of potential future schedule, budget, and quality problems during development. Additionally, the maintenance of the products may be particularly difficult.
- Small staff losses and large staff additions. This trend usually indicates that the project is behind schedule. When this trend appears, the schedule, budget, and quality would typically be in jeopardy. This trend occurs when a project is in trouble and trying to catch-up.

In general, the NAWC metrics office analyzes the above data and trends to determine if the staffing levels are as planned and that the contractor's staff is stable. Any adverse trends would be closely analyzed in conjunction with the remaining trends.

Spending Rate

To analyze the spending rate, the NAWC metrics office concentrated on the staff hours expended because, typically, the effort expended is the largest cost variable on a project. Because software development is a human-intensive and -dependent activity, the effort expended is usually the least controllable cost variable. To analyze the spending rate, the metrics office used charts similar to Figures 3-3 and 3-4. The data for these charts come from total staff-hours data on data collection form 2. The UWLs and LWLs in Figures 3-3 and 3-4 are ±10%.

Because Figure 3-3 is cumulative staff-hour data, any differences between the planned and actual curves early in the project would be small. For this reason, small but steadily increasing differences between the curves were considered warning signals. Also, because this chart is cumulative data, the NAWC metrics offi‿e would be particularly wary of trends that reverse. Here, the project strives for stability in the actual curve. The metrics office used a chart similar to Figure 3-3 to analyze the following trends:

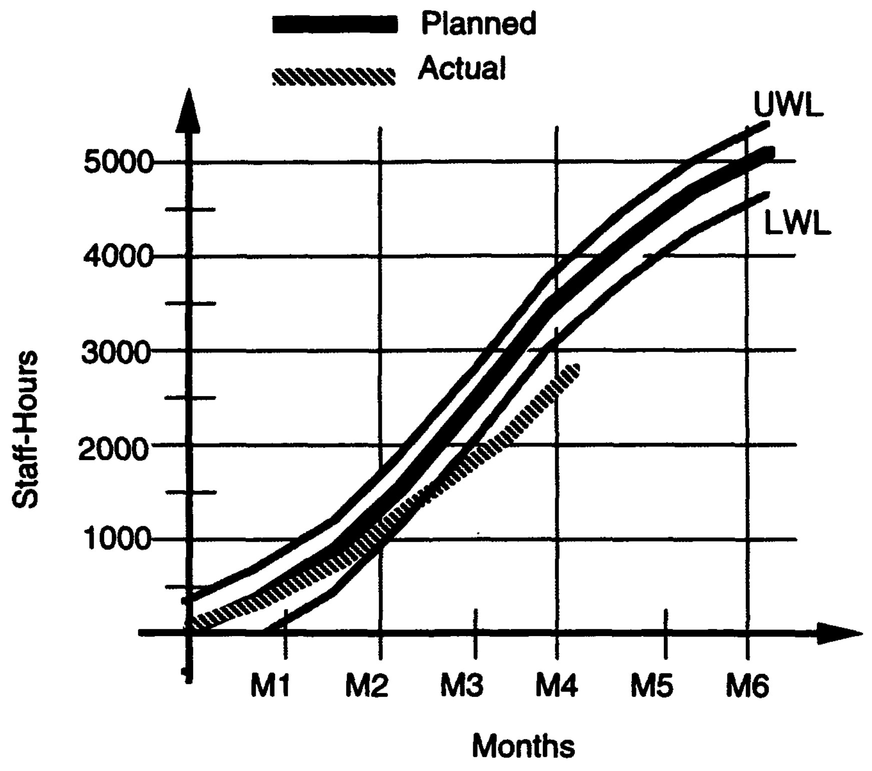


Figure 3-3 Sample Cumulative Staff-Hour Data

- Case 1: Actual curve is greater than UWL. Once the actual curve exceeded the UWL, it would be difficult to complete the project within budget. The budget might be met, though, if an equivalent rise in the work being completed was increasing at the same rate. This implied that the project would be completed within budget and earlier than planned. However, if the work is not being completed at a pace quicker than planned, minimizing the budget overrun and still completing the project on time and at a desirable level of quality would be difficult.

  The NAWC metrics office would watch carefully for a reversal of this trend. If the trend reverses, rather than stabilizes, less time will be spent on the project and now completing the project on schedule would become a risk. Sometimes, in an attempt to complete the project within budget, functionality is removed from the project. In this scenario, although the budget may be met in the near term, the project usually costs more than planned when the removed functionality is eventually developed. The NAWC metrics office would use trends from all of the remaining measures for more insight into the status of the project.

- Case 2: Actual curve is less than the LWL. In this case, the spending rate is much less than planned. Here, the NAWC metrics office would extrapolate data from the product delivery and work progress measures to estimate if the project would be completed as scheduled. Also, data from the post deployment work measure would be analyzed to determine if future work was being backlogged. A reversal of this trend would be analyzed to determine if the additional effort being expended is useful towards completing the project and/or improving the quality of the products.
- Case 3: Actual curve is less than the UWL and greater than the LWL. In this case, the budget is less of a risk. The NAWC metrics office would analyze the other measures to ensure that the progress of the project is commensurate with the effort being expended.

Figure 3-4 shows the chart of monthly staff-hour data expenditure. Because the data were monthly and not cumulative, reversal of trends is not a concern, but is actually desired when the current trend is the opposite of what is desired. If the actual curve is consistently above the UWL and the work being completed is not rising equivalently (as seen in the product delivery and work progress measure) or the backlog of work is not decreasing (as seen in the post deployment work measure), then completing the project within budget would become a risk. Just the opposite is true if the actual curve is less than the LWL; i.e., the budget is not a risk, but completing the project on time is. Fluctuations in the chart in Figure 3-3 should also be seen in Figure 3-4.

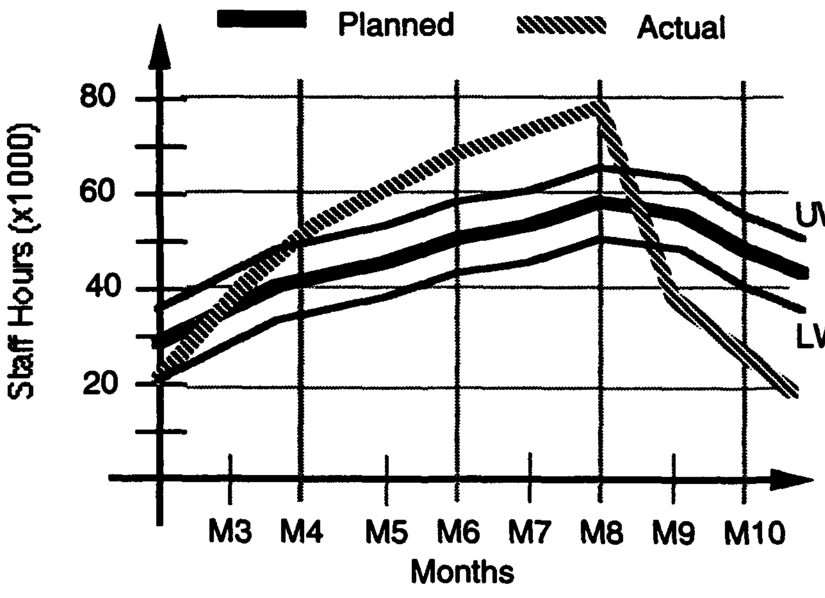


Figure 3-4 Sample Monthly Staff-Hour Data

Allocation of Resources

When determining if the proper resources are being allocated, it is important to analyze that the effort is being expended adequately by the appropriate labor categories for each functional area. To analyze the allocation of labor, the metrics office uses data from data collection form 2 to generate a bar chart similar to Figure 3-5.

For each project, a chart similar to the one in Figure 3-5 is generated each month to highlight where the effort is being expended on the project. Charts are modified to better highlight areas that may be of particular interest or risk to the program. For example, the "others" labor categories or the "others" functional areas could contain items that need to be shown separately.

The NAWC metrics office would analyze the following issues:

- Is a disproportionate number of staff hours being spent on later life-cycle functional areas that depended on earlier functional areas? For example, in Figure 3-5, if the project had not yet had its SSR (i.e., requirements review), there would be a concern that a large number of staff hours were being spent in design, coding, and integration and testing.
- Are an adequate number of staff hours being expended by SQA and configuration management (CM)?
- If the "others" categories (either the labor category or the functional area) have a larger than expected number of staff hours being expended, where and what are those staff hours being expended towards?

In general, the NAWC metrics office would analyze the chart and determine if the distribution of staff hours was appropriate considering where in the life cycle the project was.

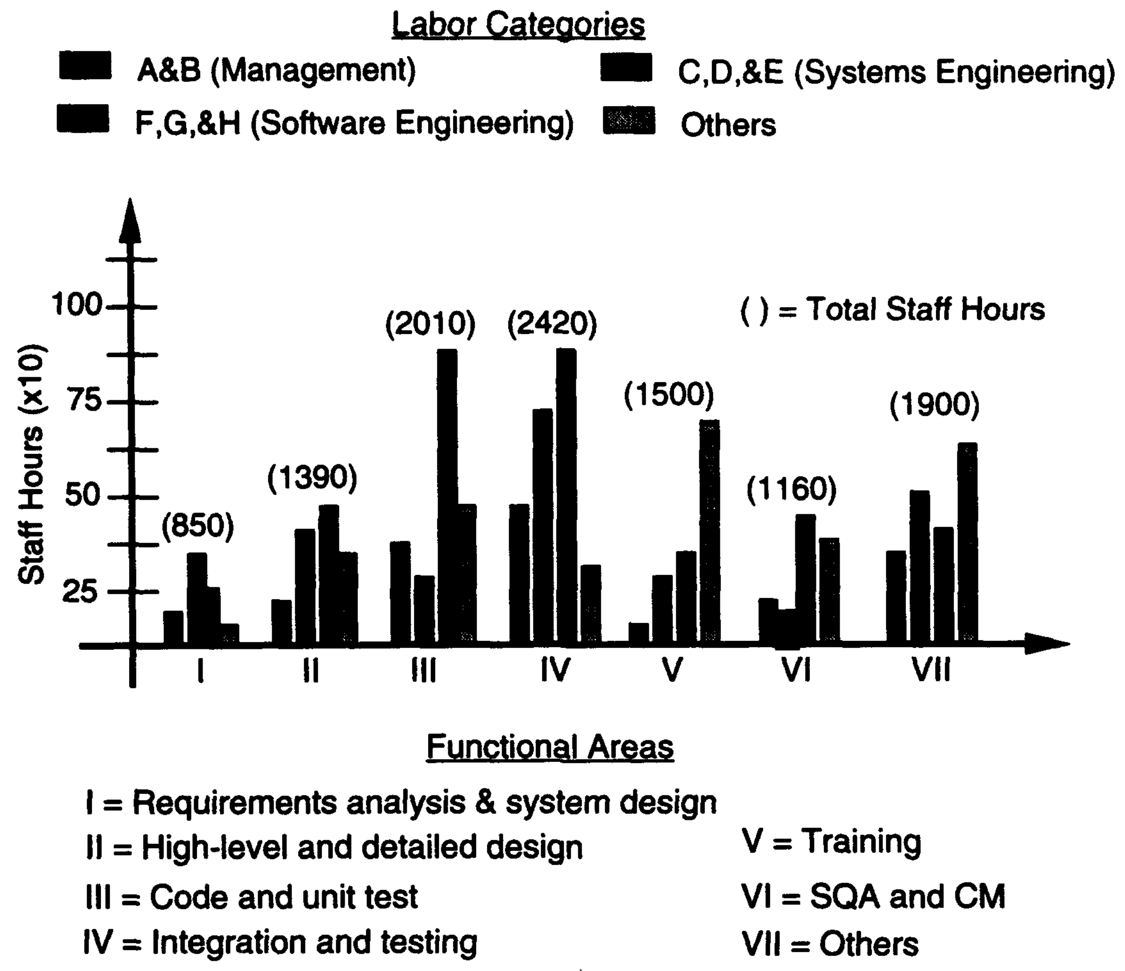


Figure 3-5 Sample Effort Distribution Chart

Additionally, each of the functional areas in Figure 3-5 should have a chart similar to Figure 3-6 to ensure that the expenditure of staff hours over time for each functional area is in line with expectations. From this chart, the NAWC metrics office would look for spikes and dips in the labor category bars. Spikes are where more effort is being expended during a particular month than was spent in the months before and after the month in question. Likewise, a dip is where less effort is expended during a particular month than was spent in the months before and after the month in question.

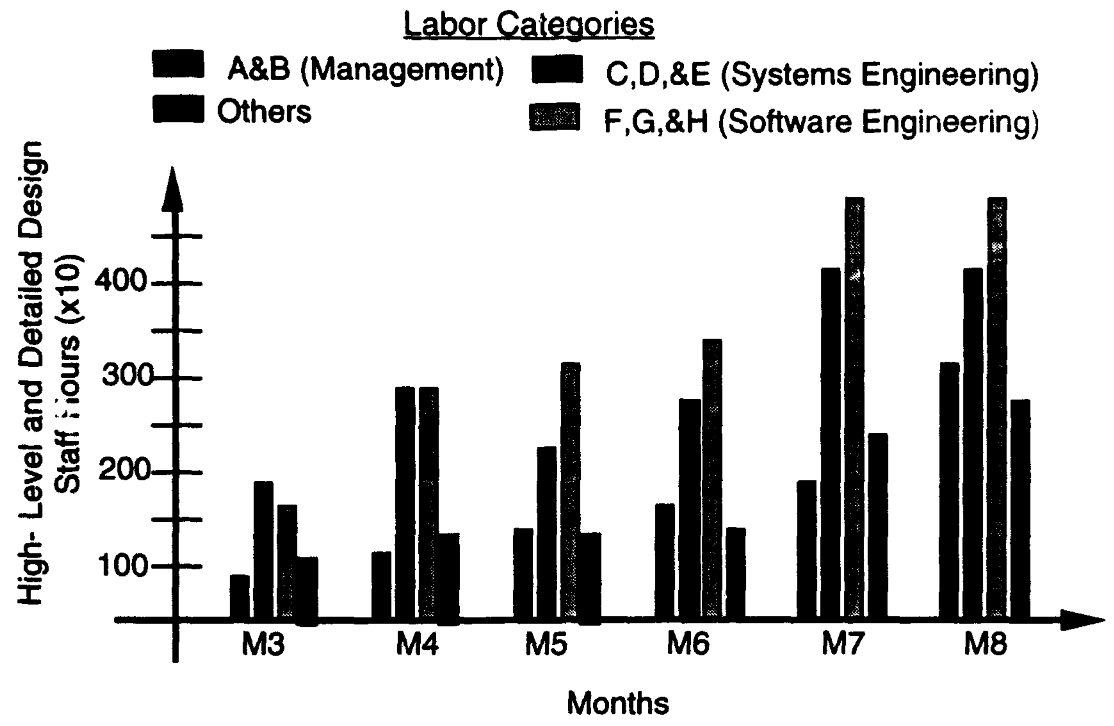


Figure 3-6 Sample Effort Distribution by Functional Area

Accuracy of Size Estimate

The accuracy of the size estimate is important because the size estimate is usually the basis for planning the amount of effort and time (schedule) needed to complete a project. An increase in the expected size is an early indicator that the budget and the schedule for completion may not be met. To track the size of the software with this type of indicator, a chart similar to Figure 3-7 was generated monthly using the data from data collection form 5.

From the chart in Figure 3-7, the NAWC metrics office would look for growth in the estimated size of the software. As actual data became available, the chart would have bars for the actual SLOC, the amount of SLOC estimated to remain, and the sum of the two (i.e., total SLOC). The total SLOC was then compared to the estimate to indicate variances.

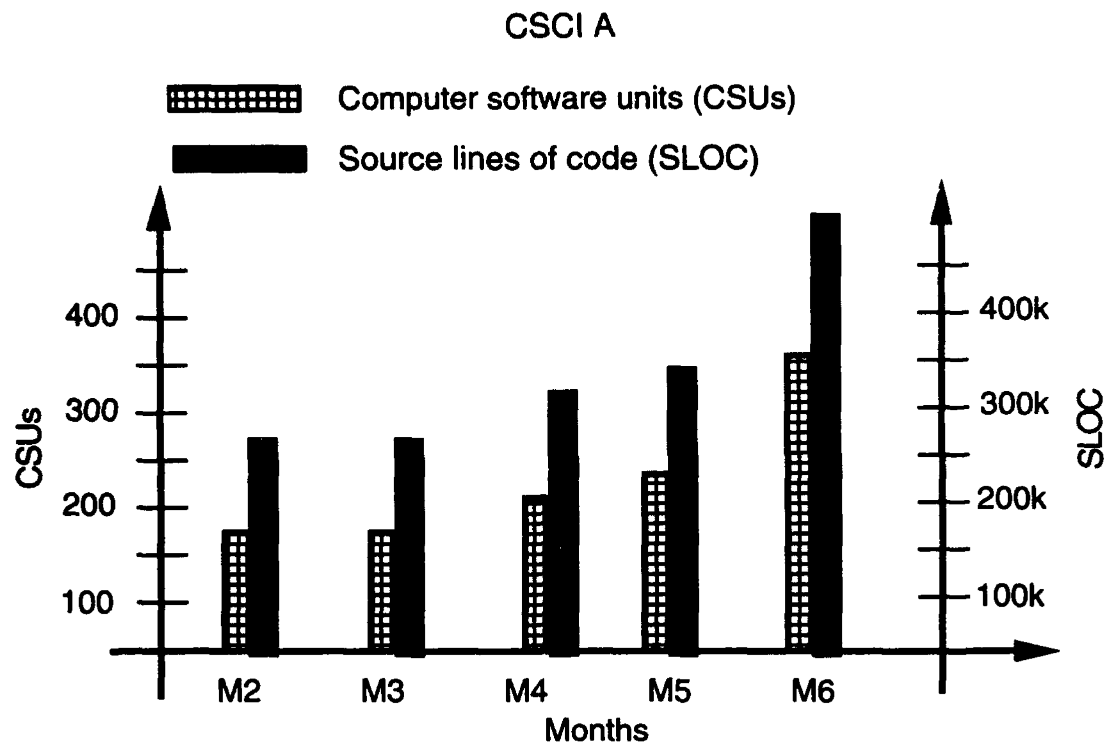


Figure 3-7 Sample CSCI Software Size Tracking

Figure 3-7 also shows the planned number of CSUs. The design, coding, integrating, and testing of CSUs are the best indicators of the amount of effort and time needed to develop the system schedule. Experiences at NAWC show that a better indicator of schedule progress was an increase in the planned number of CSUs.

Another analysis of Figure 3-7 is to determine whether there is an equivalent increase in the number of SLOC when the number of CSUs increase, and vice versa. For example, if the estimated SLOC goes up 10%, does the number of CSUs go up 10%? If not, then the program manager should investigate closely the estimation methodology as well.

3.4.2 Progress Issue Analyses

Although the progress issue is easier to articulate than the resource issue, it is more difficult to address. The issue is: will the contract be completed on schedule, and if not, when will it be completed? The measures in increment one that provided insight into the following questions on progress were:

- Schedule (milestone commitments); is the contractor meeting commitments on time and within a prescribed level of quality?
- Development progress; what is the true (objective) progress of the contract?
- Schedule and forecasts to completion; were they realistic?
- Functionality allocations; were they shifting from earlier to later builds?
- Rework; were high levels of rework having an impact on progress?

Schedule Commitments

Planning and meeting individual interim schedule commitments is the key to completing the overall project on time. The NAWC metrics office used charts similar to Figures 5-8 and 5-9 to analyze the individual commitments and determine how progress towards completing them affected the overall schedule. The data for the chart in Figure 3-8 are from data collection form 3; whereas the data for the chart in Figure 3-9 come from updates to the plans. These updates could be formal updates (i.e., those discussed and agreed to by the contractor and government) or the data could have been retrieved informally by being reported monthly along with the actual data on data collection form 3.

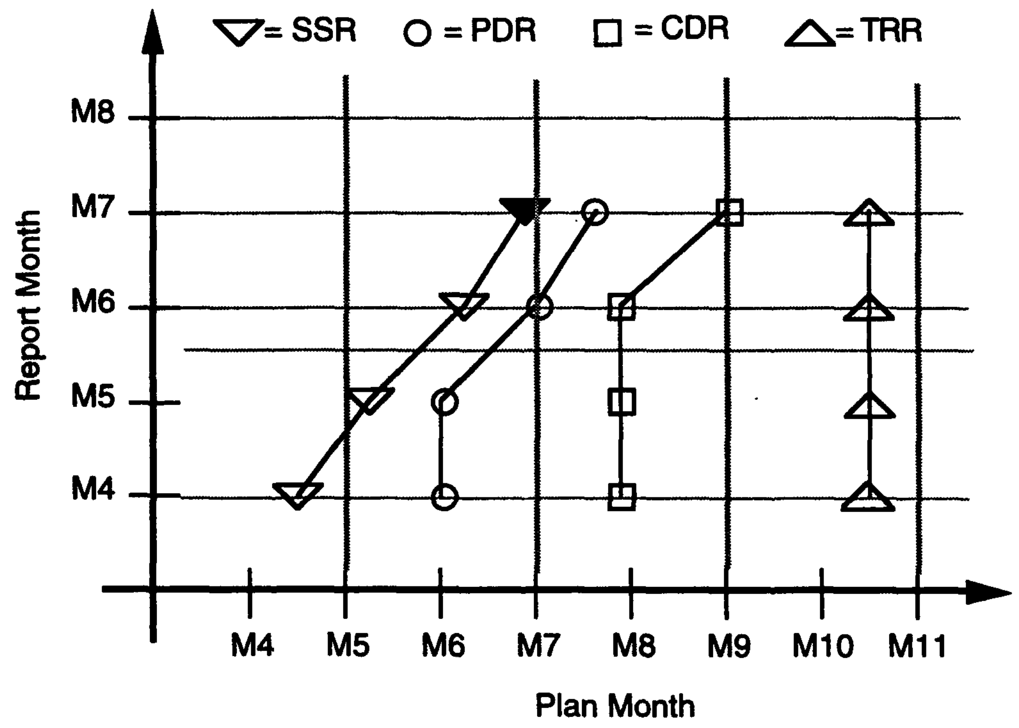


Figure 3-8 Sample Milestone Commitment and Planning

In Figure 3-8, the data from data collection form 3 is plotted monthly on the y-axis. The x-axis corresponds to the current plan that the government program office is using for managing the project. Charts similar to Figure 3-8 would be generated for the milestone data on data collection form 3 and for the product data on data collection form 3. Likewise, other intermediate milestones agreed to by the contractor and government program manager would have another similar chart.

To analyze the chart, the NAWC metrics office looks for early commitments that have slipped and determines if future commitments are slipping an equivalent amount. For example, in Figure 3-8, the SSR milestone has slipped four consecutive months before finally being completed in month 7. However, the next immediate milestone, PDR, did not slip until the third reported slip of the SSR. In month 4, the PDR was to be complete approximately one and one-half months after SSR. In the latest report, month 7, the PDR is shown as approximately a half month later. In a situation like this, the NAWC metrics office would question how SSR could slip two months while TRR did not slip at all.

The chart in Figure 3-9 can be generated from any plan data on schedule commitments or expected progress. To generate the chart, the NAWC metrics office selects key schedule items and then graphs the various plans for each item on a separate chart. Often, schedule progress is reported to be in line with the plan. By charting the various plans, the NAWC metrics office could then ask, "in line with which plan?"

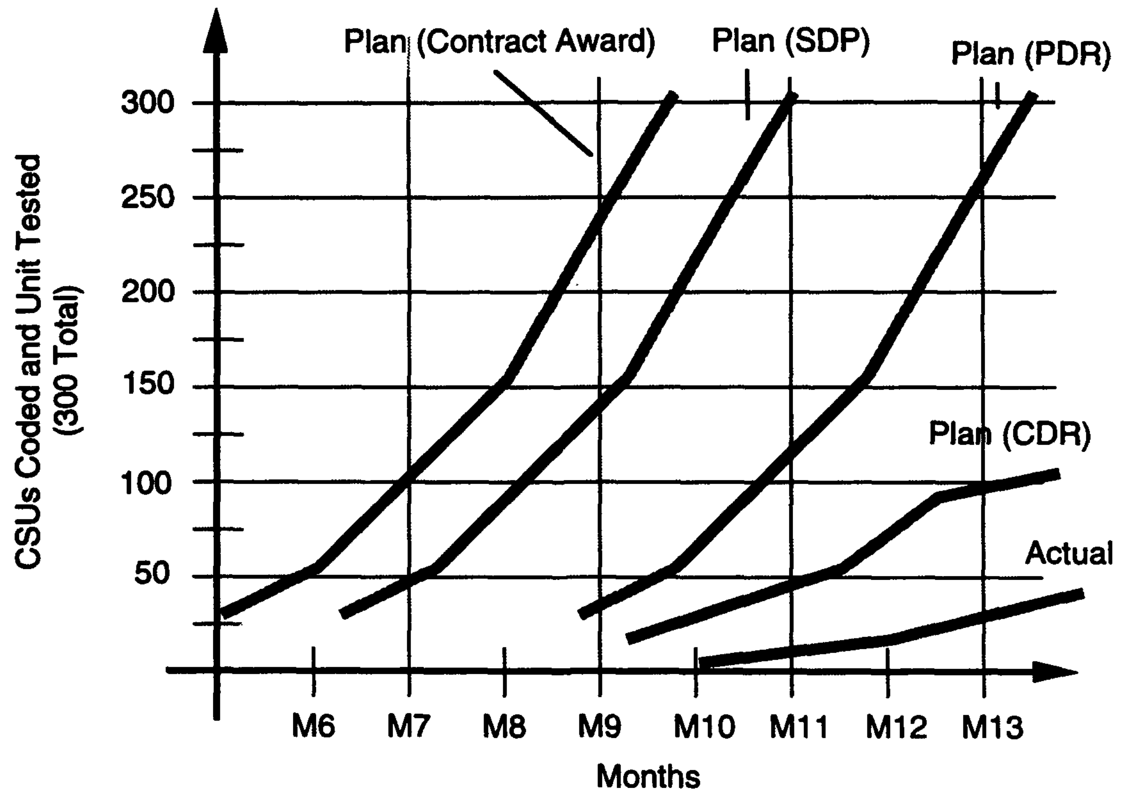


Figure 3-9 Sample Analysis Using Multiple Plans

From charts similar to Figures 3-8 and 3-9 for key program items, a project can better plan and determine when the project will be completed and what future items will slip given the slippage of earlier, dependent items.

Development Progress

Intermediate milestone completion and product deliveries are used to indicate if the project is meeting its overall schedule. In addition, charts similar to the one in Figure 3-10 give better insight into the progress of completing the work to be done. A chart similar to Figure 3-10 would be generated, when possible, for:

- Requirements related work items; with curves for requirements in the SRS, requirements documented in the SDD/IDD, and requirements tested in FQT.
- Work items related to the CSU progress; with curves for CSUs designed, CSUs coded and tested, and CSUs integrated.
- Testing related work items; with curves for regression tests, integration tests, and FQT tests.

The data for each of these charts come from data collection form 4. The plan data are always from the currently approved plan. Additionally, each curve could be generated as a separate chart with a UWL and LWL. If generated, the UWL and LWL are generally around ±10 or 20% depending on the data item and the issue it is measuring.

To analyze a chart similar to Figure 3-10, the NAWC metrics office would look for variances between the actual and planned curves of the work items. The work items all have defined exit criteria that must be met before the work item can be included as data for the actual curves. If the variances are small, the NAWC metrics office would validate the data being included in the charts to ensure that the work items being reported as complete are meeting the exit criteria.

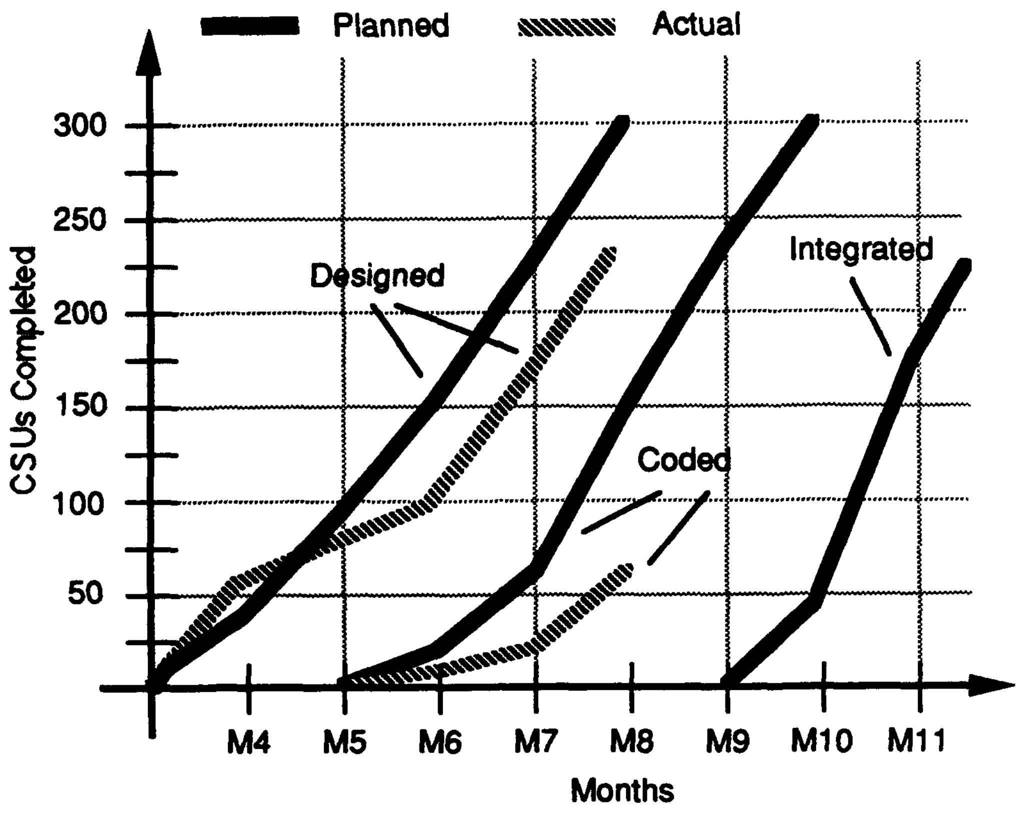


Figure 3-10 Sample Indicator for CSU Development Progress

Forecasts to Completion

The NAWC metrics office uses software measurement data to forecast when the project would be completed. The forecasts use data from data collection forms 3 and 4 to generate a series of charts similar to Figure 3-11. A chart is normally generated for:

- Milestones to be met.
- Products to be delivered.
- Work items to be completed.

The estimated slips in the charts are determined from the data collected or the information provided by analyzing the development progress and schedule commitment issues above. Additionally, information on the allocation of resources, staffing stability, testing SPRs, functionality allocations, and effort expenditures are all used to determine the forecasts.

At a minimum, the forecasts for the milestones would be equal to the amount of slippage seen in early milestones. Usually, additional information, including historical data when available, would also be used to generate a forecast. A similar approach was taken for the products to be delivered.

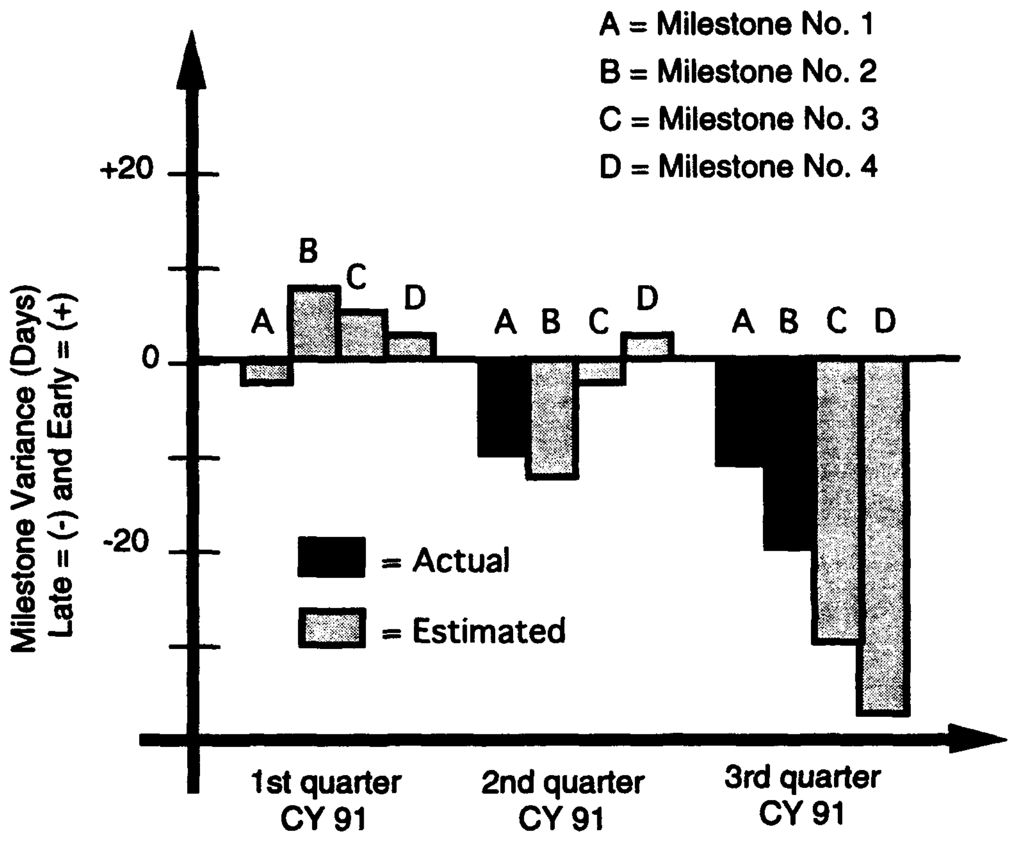


Figure 3-11 Sample Milestone Completion Variances

Functionality Allocations

The NAWC metrics office used a chart similar to Figure 3-12 to reveal the postponement of functionality from early builds to later builds. Sometimes, such postponements were made to achieve early schedule completion successes. Typically, this translated into overall budget and schedule risks (and maybe a quality risk as well).

A chart similar to that in Figure 3-12 was generated for the project. The data for the chart come from summing the CSU data shown on data collection form 5 for each CSCI.

To analyze the chart. the NAWC metrics office would look for:

- Decreases in the number of CSUs planned in early builds.
- Increases in the number of CSUs in later builds.
- The addition and deletion of entire builds.
- Growth in the number of CSUs planned for the project.

Of course, each trend was correlated to other measures. More importantly, when the number of CSUs increased or decreased in builds, the metrics office would determine if planned staffing levels and schedules also changed.

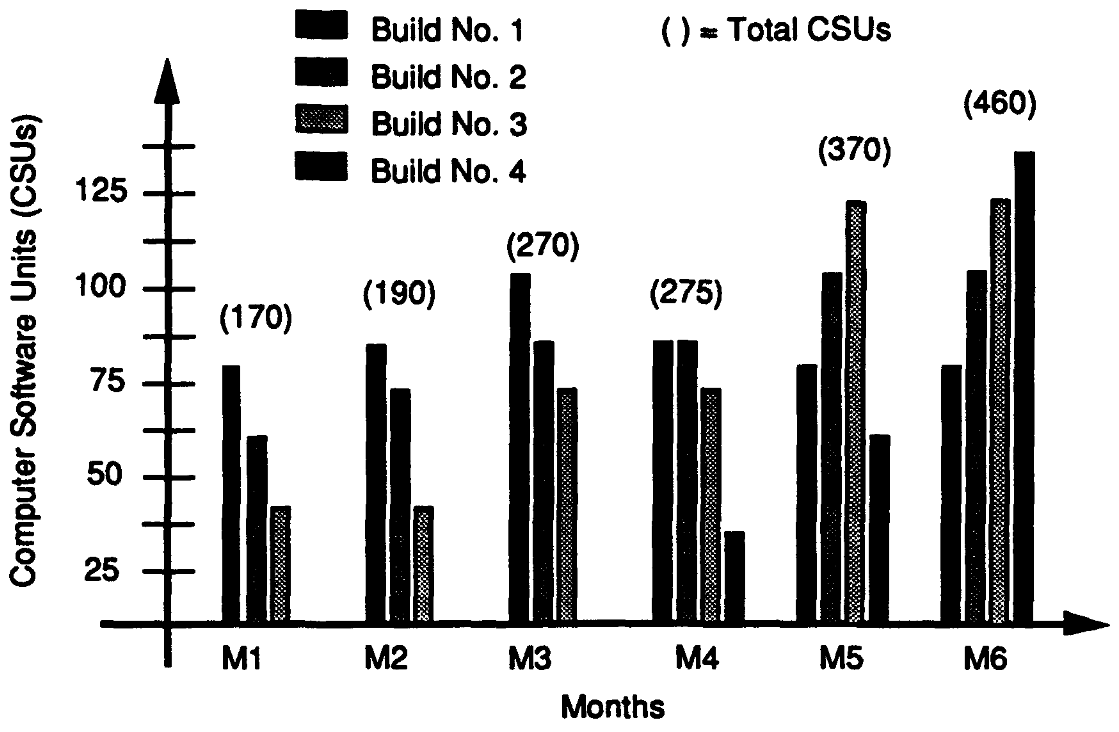


Figure 3-12 Sample Functionality Allocation Plan

Progress Impacts Due to Rework

The greatest impact on progress and completing the project on schedule, is the unplanned time needed to rework products that were defective. Unlike a lot of industries where defects are found and the entire product might be discarded, software is not easily discarded;instead, defects in software must be fixed. To correct a defect properly, not only does the code itself need to be corrected and retested, but the documentation needs to be corrected back to the point where the defect originated. (For example, if the defect was a design error, then the documentation for the design and testing, along with code, needs to be corrected.)

Obviously, the NAWC program manager would have liked to have defects reported to him/her as early as possible. However, because of the nature and environment of contracts, that type of information could not always be obtained. Increment one analysis, therefore, assumed that the code had been delivered and some level of formal testing started when the project obtained its first insight into the quality of the software. However, if possible, data on walkthroughs, inspections, audits etc., would be obtained and used in a similar manner.

To determine if unplanned rework has or will have an impact on progress, the NAWC metrics office used charts similar to Figures 3-13 and 3-14. The data for the charts came from data collection form 6.

To analyze the data in Figure 3-13, the metrics office would look at the net change in the backlog of SPRs. At a minimum, the priority 1 SPRs were shown separately as in Figure 3-13. Another, useful chart generated sometimes from this data was an x-y graph with the net changes shown on the y-axis and time in months on the x-axis.

Type of SPR	Reported This Month	Closed This Month	Open Beginning of Month	Open End of Month	Net Change
Priority 1	3	2	1	2	+1
Others	23	9	15	29	+14

Figure 3-13 Sample Backlog Report of SPRs

The NAWC Metrics Office also analyzes the monthly overall trend of SPRs reported, closed, and remaining open similar to Figure 3-14. It was helpful to have a chart for each of the test sources identified on data collection form 6 as well as a chart for the entire project. With these charts, the NAWC metrics office was looking for:

- The slope of the total reported curve to level off.
- The curve for the number of SPRs closed moving towards the curve for the number of SPRs reported.
- The curve for the number of open SPRs trending toward zero.

However, such trends were not necessarily a sign that the quality of the products was stabilizing; instead, it could be a sign that the search for defects had diminished. This is a natural occurrence. Sometimes when many SPRs are backlogged, activities that uncover defects, e.g., testing, are reduced while activities to correct defects are increased. To determine if such a situation existed on a project, the NAWC metrics office looked at where effort was being expended (e.g., in testing or in correcting known problems) and also correlated the trend with the curves for the testing work items in the work progress metric.

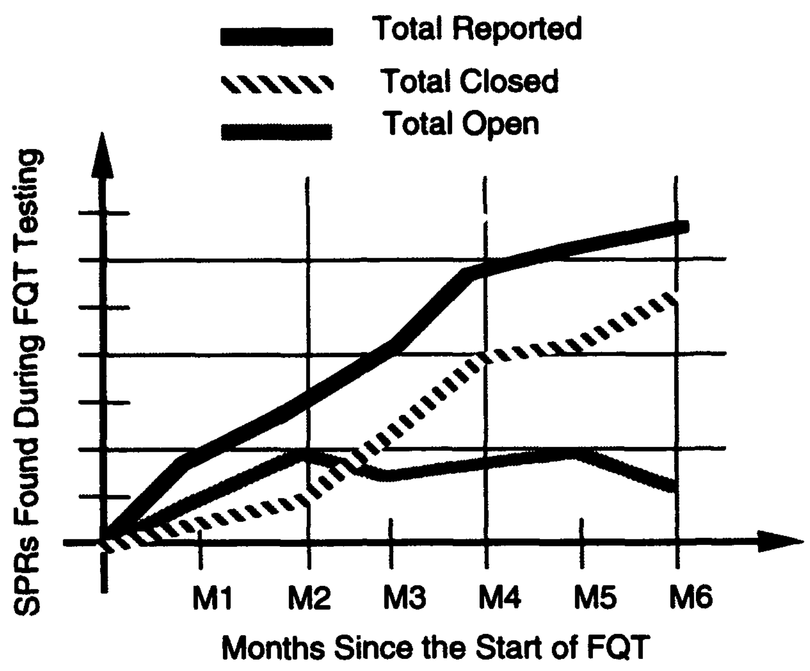


Figure 3-14 Sample Progress Delays Due to Open SPRs

3.4.3 Analyses of Technical Quality Issues

The technical quality issues cover the quality of products being produced. The primary concern of the technical quality is that interim products are stable and complete so that successive processes using those products do not compound mistakes and thereby drive up cost through added rework. The measures can provide insight into technical quality issues such as:

- Problem reports (e.g., severity of problems, number of problems, timely resolution of problems, or high density levels of problems in products or processes).
- Product stability and volatility (e.g., requirements, design, and functionality allocations).
- Rework (e.g., later processes finding defects caused by earlier processes).

SPR Densities

The NAWC metrics office used a chart similar to Figure 3-15 to determine the density of problems found within the code for a project. Software problem reports (SPRs) are always expected. Likewise, a project would probably never discover all errors; "testing does not ensure the absence of errors, only their existence." Discovering more errors than expected, though, may be an indication of poor software and potential future problems. When too few SPRs are found, it is possible that the project was not effectively looking for them (e.g., testing).

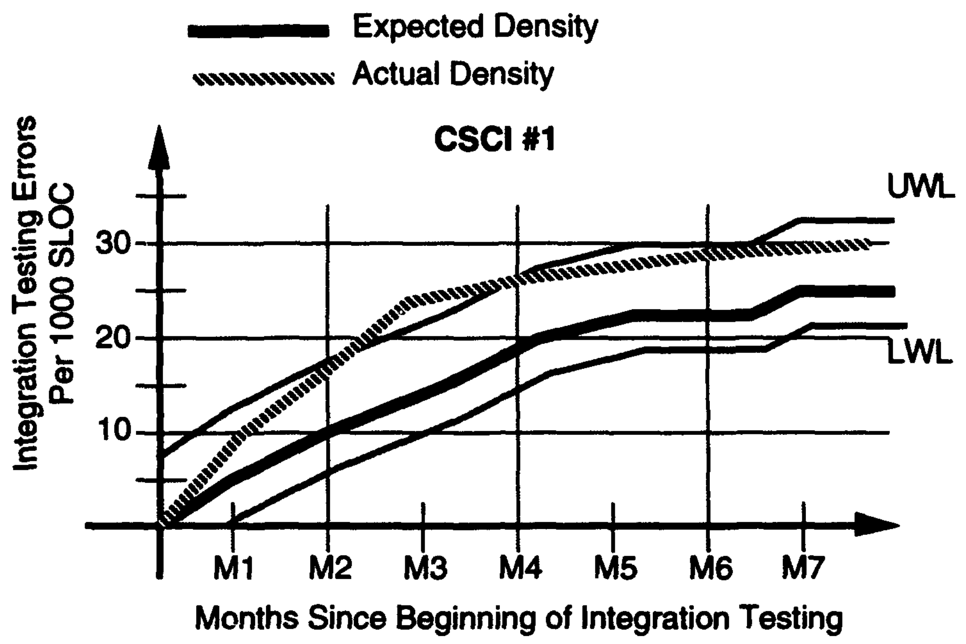


Figure 3-15 Sample SPR Density

In Figure 3-15, the data for the y-axis are the result of the total SLOC divided by the total number of SPRs. Actual data come from data collection forms 5 and 6. Typically, the tracking of SPR density starts after formal testing. However, when earlier error data are available (e.g., data from inspections, walkthroughs, and audits), a chart for each is generated also.

The UWL is +25% and the LWL is -10% of the expected curve. The expected SPR density curve is subjectively determined and dependent on the key factors and goals of the project. Criteria used to determine the expected number of SPRs includes:

- Historical data from past similar projects. The best method to determine how many SPRs should be expected for a project is to analyze SPR density of past similar projects.
- Criticality of the mission and the potential losses if the software fails. Depending on the criticality of the mission, more or less rigorous development and assurance methods may be applied in earlier development phases. For example, rigorous inspections and walkthroughs may be performed during requirements analysis and again during design if the system is life critical or requires a very high reliability. Dependent on the criticality, the level of testing may be more or less rigorous also; more rigorous testing will supposedly uncover more SPRs prior to delivery.
- Complexity and precedence of the system. The overall complexity of the system and whether or not similar systems have been built before will also have an impact on the number of problems that occur and are uncovered. The more difficult the system, the more errors that should be expected.

To analyze the SPR density charts, the NAWC metrics office primarily looks for trends away from the expected curve, always remembering that the expected curve was subjectively determined initially. The analysis is coordinated with other measures such as:

- Effort measure: to ensure that effort is being expended toward finding errors.
- Work progress measure: to determine what level of testing is being performed. Here, the NAWC metrics office looks at the number of tests being performed in relation to the number of requirements tested, CSUs existing, etc.

Product Stability and Rework

The stability of products is determined by the quantity and existing backlog of SPRs for the project and the individual products. Individual products that have many SPRs reported (i.e., regardless of whether the SPRs were open or closed) should be investigated, and possibly redeveloped given the criticality of the mission for that product and the system.

One stability issue is the total volume of SPRs reported for a product. Increment one measures focused on the number of SPRs received for each CSCI. For each CSCI, a chart similar to Figure 3-16 was produced. This chart would include a bar for the number of SPRs for each module or CSC within a CSCI. Charts such as that in Figure 3-16 are commonly called Pareto charts. They were used to determine where the majority of the errors in a product were occurring. By determining where most errors occurred, the NAWC metrics office would recommend more attention and scrutiny to those CSCs. This could be accomplished by requiring more testing (or broader testing), recoding the module to optimize it, or totally redeveloping the module.

The data for the chart in Figures 3-16 and 3-17 come from data collection form 6. Preferably, the NAWC metrics office has the SPR data (or access to it) for the SPRs for each CSC within a CSCI. If not, then the chart is produced at the CSCI level.

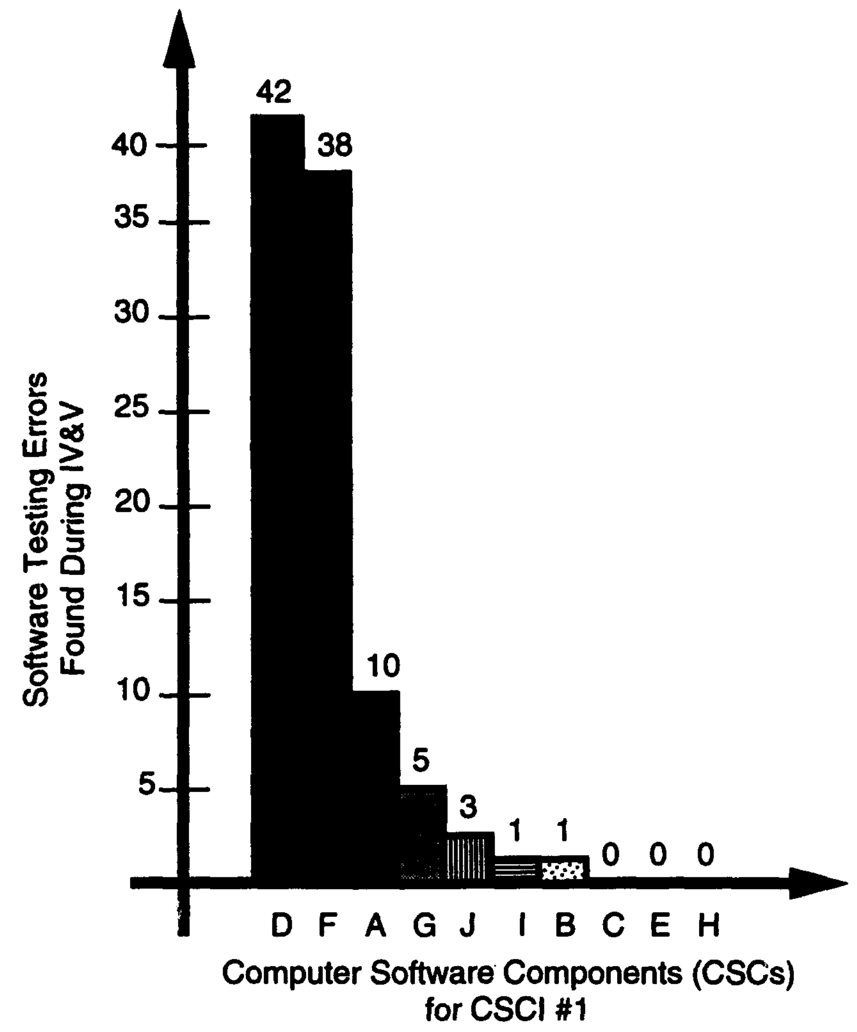


Figure 3-16 Sample Pareto Analysis of CSCI Defects

For further analysis, it is likely that additional data will be needed. The additional data could be module defect densities, module complexities, etc. Because there could be many reasons for more errors occurring in some modules as compared to others; it is difficult to predetermine what extra data are needed for analysis, and requiring data for all possibilities would not be cost effective. For example, the complexity of the modules may be higher in the more error-prone modules; less experienced personnel may have developed the module; personnel added after requirements and design may have coded the module without fully understanding the module's mission; the module may have been the last completed and done in a hurry to meet the schedule; the total size of the module may have been a hindrance; the mission of the module may not have had a precedent; or the module may interface with a complicated set of external hardware. The list of reasons can be long, but whatever the reason for the errors, the NAWC metrics office would suggest extra attention to those modules with the most errors.

The metrics office would also produce a chart similar to Figure 3-16 showing where the errors were being discovered in the process. This was accomplished by producing a bar (x-axis) for the number of SPRs found for each testing process (e.g., IV&V testing, FQT testing). To analyze these data, the NAWC metrics office looked at the total SPRs found, and the number of SPRs found in the earlier testing processes compared to the number found in the later processes. In this case, it was desirable to have the earlier processes finding the majority of the errors. If a large number of errors were being found later in the process, then the NAWC metrics office would recommend more and better testing in the earlier processes.

In summary, the NAWC metrics office would analyze a chart similar to Figure 3-16 to determine where the majority of errors were occurring and where in the process they were being uncovered. Another stability problem is the number of SPRs backlogged. To determine this, a chart similar to Figure 3-17 is produced. Here, the NAWC metrics office would be concerned with the amount of work that was being backlogged in the form of rework that needed to be completed before the completion of the project. Excessive backlogs could become budget and schedule risks, especially when little or no rework was budgeted in the effort and schedule planning. Also, an excessive number of high-priority SPRs could significantly reduce a system's mission capability. The key analyses were to look at the total SPRs open by priority and the length of time the SPRs were open. If SPRs were open too long, then other functionality dependent upon them may also suffer.

Priority Levels	Number of SPRs That Have Been Open x Days				Totals
	$x \leq 30$	$30 < x \leq 60$	$60 < x \leq 90$	$x > 90$	
Priority 1	2	1			3
Priority 2	3	1	1		5
Priority 3	3	2	1	1	7
Priority 4	12	9	6	5	32
Totals	20	13	8	6	47

Figure 3-17 Sample SPR Longevity Report

3.5 Reporting the Analysis Results

The process to report data analysis results is considered the flow of information from the NAWC metrics office to a government program manager who would use the information to help manage his or her project.

There were two portions to the analysis process: (1) a data validation portion that used input from the contractor to validate the findings, and (2) the presentation of the results to the program manager.

For the data validation portion, the metrics office used the results from the analysis process and developed a list of questions and/or risks that it would postulate from the data trends. It first would take these questions and risks to the contractor responsible for developing the software and go over questions with the contractor. This first step in validating the data was to ensure that the metrics office had accurate and reliable data and to address some of the trends to determine whether or not they were possible risks. The metrics office then used additional information obtained from the contractor to develop an analysis report for a project that would be presented to the program manager.

The analysis report was not intended to be used as a stand-alone report; it was merely to be used to summarize the overall performance regarding predetermined issues. In fact, because the analysis process was intended to be dynamic, the feedback of results using a single report was impractical, illogical, and rarely used. Ideally, the NAWC metrics office would have a program review with the program manager where the analysis report was briefed along with other key indicators.

The measurement data reported and collected by the NAWC metrics office was always treated as proprietary and confidential. Because of these process requirements, the metrics office would provide feedback to the program manager or office regarding data on its project only. Feedback to other required parties, including NAWC personnel, would only be done anonymously. That is, projects would not be identified in analysis reports from the metrics office, except to the government program manager or office for the project from which the data were collected or reported.

4. Piloting the Processes

As mentioned earlier, after developing a process, the SMPAT tested each process before recommending that the process be implemented throughout NAWC.

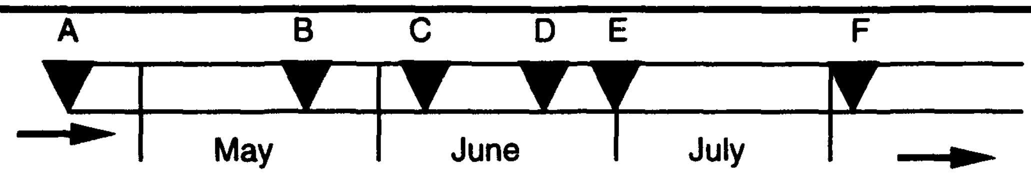
When initially searching for a project to pilot test the processes, the SMPAT determined a set of attributes that it wanted the project to contain. Although NAWC does not have a single, typical project that represents any set of attributes, the SMPAT felt it needed a set of criteria so that the chosen project could adequately test the process. The desired attributes included:

- Coded in Ada for the majority of the software.
- Developed to DOD-STD-2167A guidelines.
- Contained between 100,000 and 500,000 lines of code.
- Had a clause in the contract to collect measurement data.

The Ada and 2167A attributes were needed because Ada and 2167A represent the DoD's desired development methodology. The size attribute was established because the SMPAT did not want a project that was too small and that could invite criticism that the pilot test results would not scale up. Nor did the SMPAT want a project so large that the mere complexity due to its size would swamp its efforts. It also wanted to pick a project that already had software measurement included in the contract so that it could eliminate or greatly reduce possible legal and contracting issues.

Once the pilot was chosen and agreed upon by all parties, a plan was laid out that went from a kickoff meeting to the first data collection cycle. A brief timeline of that plan is shown in Figure 4-1. The plan had milestones for: a kickoff meeting, a meeting with the contractor, production of a draft and a final report documenting the data collection process to be followed, and the beginning of the first data collection cycle. The SMPAT met each milestone on the original schedule. This greatly enhanced the SMPAT's credibility with the contractor and its Navy sponsor.

The biggest obstacle for the SMPAT was learning the process and terminology of the pilot project. The SMPAT then had to translate that terminology and find the hooks in the contractor's process to capture the data it needed. This learning process became a major coordination effort. The SMPAT had to communicate and coordinate its efforts with the NAVAIR program office, the NAWC IV&V agent, the prime contractor, the NAVAIR configuration management subcontractor, and the IV&V agent's project management consultant.



A = (April 10)	Kickoff meeting to familarize SEI with project (SEI and NAWC)
B = (May 22)	SEI delivers pilot data collection guide
C = (June 2)	SEI and NAWC review pilot data collection guide
D = (June 23)	Review meeting to present the data collection process (SEI, NAWC, and contractor)
E = (July 1)	Start first data collection cycle
F = (August 3)	Data received from first data collection cycle

Figure 4-1 Pilot Project Timeline

The result of the pilot planning was a special report describing the details of the data collection process that the SMPAT was to use. The data collection guide specifically described what data were to be collected and how they were to be collected—in the project and contractor's terminology and within the bounds of the processes already in place.

The SMPAT goal was not only to test the process in the NAWC software measurement guide, but to enlist complete cooperation of the prime contractor, without costing the program office, the NAWC IV&V agent, or the prime contractor any additional money not already allocated to software measurement.

The SMPAT met all of these goals. It was able to collect all data described to support the measures in increment one. As it turned out, the prime contractor had just started to develop its own software measurement program. So, the prime contractor was enthusiastic about working with the SMPAT and learning what measures and data its customers were interested in collecting. The data collection guide identified what every piece of data was and how it was to be collected. The NAWC Software Measurement Guide showed how the data collected were going to be used. Both documents further assured the contractor that there would not be abuses of data collected in its project.

Regarding the cost, the NAWC Software Metrics Office collected all data from existing CDRL items. Other than some informational exchange meetings, there was little or no additional effort required on the part of the program office, the NAWC IV&V agent, NAWC subcontractors, or the prime contractor.

Even though the SMPAT met all of its goals with a minimal additional cost, it still had some difficulties. As expected, there were translation problems in determining how the contractor was counting SLOC. Here, though, terminology was the problem. For example, the SMPAT defined reused code as any code that was delivered, existed in a previous release, and was not modified. But the prime contractor was not counting reused code using this method.

When a CSU was modified, even if only one line was changed, the entire CSU was counted as modified. Both definitions seemed logical, but the contractor who had to do the counting chose a more practical implementation. Without a sophisticated set of tools and an equally sophisticated process to accompany them, the SMPAT method might not have been practical.

Another problem encountered was the laborious task of deciphering the cost accounting data and trying to write macros to partition the data into the needed formats. All of the effort and staffing information needed was included in the cost reporting system; it was the extraction of the data that became tedious and time consuming.

The pilot project completed a full cycle of its measurement process for increment one measures by having a feedback session with the program manager on the data analysis results. As described earlier, the metrics office met with the contractor and reviewed the results. From this meeting, additional information was gained that went into a final briefing to the program manager. The feedback session was not a private session, but was actually a larger program management review where the metrics office had a slot on the agenda. This adjustment worked quite well and is now the metrics office's preferred method.

5. Software Metrics Office and Database

5.1 Software Metrics Office

As stated earlier, NAWC established a software metrics office. The purpose of the NAWC software metrics office was to provide a focal point for the collection, analysis, and feedback of the organization's and project's software data. Before it provided the data to the general NAVAIR management, the NAWC metrics office sanitized the data to protect the identity of the projects from which it came and to respect the confidentiality of (possibly) contractor-sensitive data. Of course, projects that supplied data always had their results reported to them and compared to an aggregate of other NAVAIR projects when aggregates were available.

The metrics office had two primary functions. The first was to operate as a service group to navy program offices that want to use a software measurement process to improve their project management practices. As a service group, the metrics office tailored its methods to the contractor's development process and the contract that the project operated under. In this role, the metrics office defined, collected, analyzed, and delivered feedback to the program office. The metrics office's other function was to provide consultation to other projects that wanted information on how to use software measurement, but might not want to enlist the services of the metrics office for one reason or another. The office could consult directly by providing guidance on how to include software measurement into new contracts. The office also planned to consult indirectly thorough training and workshops on software measurement or by recommending new emerging measures to meet a project's unique needs.

In addition to these primary functions, the metrics office also developed and maintained the software measurement database and established standard terminology for different projects. The central database avoided duplication of effort and provided a stable platform for historical software information and for research into better methods and processes for developing, maintaining, and measuring NAWC software products. The data and information contained in the database are not available to others outside of the metrics office, nor are there plans to make them available.

A further incentive to establish an organization-wide software measurement office was to facilitate the translation of terminology to maximize the usefulness of the measurement process across projects. The common question was: Can the SMPAT or NAWC metrics office develop common definitions of a module, a task, a CSCI, a CSU, a program, a unit, etc.? And, did it need to?

5.2 Measurement Database

During the early planning stages of the initiative, the SMPAT recognized the strong need for a measurement database. The metrics office, along with the SMPAT, was made responsible for creating and testing a database for storing software measurement data. To keep the database as simple as possible and to get a prototype in place in time to store incoming data from the pilot project, the metrics office developed the database using a popular, commercially available spreadsheet.

The database was actually ten separate spreadsheet files. But, because the software was able to link spreadsheets transparently, the system operated as a single database. Separate spreadsheet files were created and maintained for data on:

- Staff (full-time and part-time staff head counts and full-time staff added and lost).
- Effort (staff hours by labor category for each functional area).
- Size (CSUs and SLOC).
- Milestones (planned and actual dates of major program reviews).
- Delivery (delivery dates of interim events, e.g., delivery of the SDD).
- Progress (planned and actual data on work items).
- Total defects.
- Open defects.
- Defects per CSCI.
- Defects per test source.

Raw data were either entered directly into the spreadsheets or macros were developed to extract data from other electronic files (as in the case where staff-hour data were received electronically from contractors).

Analysis of the data was done using yet more spreadsheets. First, rough data analysis and calculations were executed using macros. For example, after the spreadsheet for staff data extracted its information from the cost accounting data, then, in a separate spreadsheet, additional macros determined staff information such as added personnel, the number of personnel lost, and the number of full-time and part-time personnel used. Typically, the analysis of each issue in the analysis section had a separate analysis process consisting of one or more additional spreadsheets. Although this might appear to be "spaghetti" code and quite complicated, the design was actually top-down, building on the basic data items, and each issue analyzed was architecturally independent of other analyses. Due to the power of today's commercially available spreadsheets, particularly the ability to link multiple spreadsheets, the implementation, although difficult to explain, was actually quite simple.

Figure 5-1 lists the indicators that would be generated electronically each month for each project having data collected for it. These indicators fo [illegible] baseline set that were then used to initiate analysis of the data. From this initial analy [illegible] er indicators may be generated from the data on an as needed basis.

Indicator Title	X-Axis	Y-Axis	Indicator Type
1. Milestones	Milestone plan date	Report date	x-y
2. Product delivery	Deliverable plan date	Report date	x-y
3. Cumulative staff hours	Report date	a. Planned staff hours	x-y
		b. Actual staff hours	x-y
4. Staff stability	Report date	a. Full-time staff	Bar
		b. Part-time staff	Bar
		c. Full-time staff added	x-y
		d. Full-time staff lost	x-y
5. SPR status	Report date	a. SPRs opened (monthly)	Bar
		b. SPRs closed (monthly)	Bar
		c. SPRs reported (monthly)	Bar
		d. Total SPRs open	x-y
		e. Total SPRs closed	x-y
6. SPR/SCP Longevity report	SPRs open X days	a. SPR priority	Table
	SCPs open X days	b. SCP priority	Table
7. CSUs designed	Report date	a. Planned (monthly)	Bar
		b. Designed (monthly)	Bar
		c. Planned (cumulative)	x-y
		d. Designed (cumulative)	x-y
8. CSUs coded and unit tested (CUT)	Report date	a. Planned (cumulative)	x-y
		b. Planned, Rev. X (cumulative)	x-y
		c. CSUs CUT (cumulative)	x-y
9. CSUs coded and unit tested (CUT)	[illegible] date	a. Planned (monthly)	Bar
		b. Planned, Rev. X (monthly)	Bar
		c. CSUs CUT (monthly)	x-y
10. CSUs integrated	Report date	a. Planned (monthly)	Bar
		b. Planned, Rev. X (monthly)	Bar
		c. Designed (monthly)	Bar
		d. Planned (cumulative)	x-y
		e. Planned, Rev. x (cumulative)	x-y
		f. Designed (cumulative)	x-y
11. Monthly staff hours by functional area	Report date	Curve for each functional area	x-y
12. Cumulative staff hours by functional area	Report date	Curve for each functional area	x-y

Figure 5-1 List of Basic Indicators Produced Monthly (Continued on next page)

- continued -

Indicator Title	X-Axis	Y-Axis	Indicator Type
13. Staff hours by labor category	Report date	Curve or bar for each category	x-y/ bar
14. SPRs by CSCI	Report date	Cumulative curve for each CSCI	x-y
15. SLOC add/mod.	Report date	a. Planned (monthly)	Bar
		b. Planned, Rev. X (monthly)	Bar
		c. Designed (monthly)	Bar
		d. Planned (cumulative)	x-y
		e. Planned, Rev. x (cumulative)	x-y
		f. Designed (cumulative)	x-y
16. SLOC by language	Report date	a. Monthly for each language	Bar
		b. Total for each language	x-y
17. CSUs per CSCI	Report date	Bar for each CSCI (monthly)	Bar
18. CSUs per build	Report date	Bar for each build (monthly)	Bar

Figure 5-1 List of Basic Indicators Produced Monthly - continued

Without question, the design was not optimal, but the database prototype was able to be produced quickly and inexpensively, and it served NAWC needs for the pilot project. As the metrics office grows and additional capacity is needed, it is not known whether or not the spreadsheet-style database will continue to be used.

6. Lessons Learned

Over the course of developing the measurement program, many varied lessons were learned and some advice and postulations that had been made by others were confirmed. Some of these lessons include:

- **Start with issues, gather manager's questions related to the issues, then develop measures that help address those questions.** This was diametrically opposed to picking "a few good or interesting metrics." The SMPAT found it very helpful first to understand what the intentions of management were and what problems they faced before determining what measures to use or develop. Still, the measures in increment one were common to most software measurement initiatives. The payoff for tracing the early data definitions to management issues was realized during the analysis process. During the analysis process, having the traceability from issues to measures helped by providing the architecture for what needed to be addressed by the analysis process versus haphazard analysis to "see what we can see." Having the traceability to the issues and questions also helped alleviate some early contractor concerns of abusing the data. Showing the complete traceability from issues through to the analysis process helped build a sense of trust and willingness to cooperate with the contractor.
- **Data definition must be rigorous and unambiguous.** A well-defined data set helped throughout the process. By being able to articulate what the data collected represented, many questions and inconsistencies with other efforts that also had a form of data (e.g., configuration management) were answered in a way that built credibility for the NAWC measurement program rather than destroying its credibility.
- **The data collection process must be very specific and well defined.** Possibly even more important than how the data were defined and what they represented was how the data were collected. The NAWC measurement process specifically addressed how and where each data item was collected. This helped the process in two ways. First, those who might have resisted saw that data were being extracted from common CDRL items and, therefore, they had little basis to question the integrity of the data. The process was also helped by having common CDRL items identified as data sources. The measurement program was no longer an extraneous requirement, but instead augmented the existing process with little or no extra effort, yet, added tremendous value to the process.
- **Even for simple measures, data collection and analysis were difficult.** After about three months of planning and designing the pilot test for the NAWC software measurement processes, an SMPAT member remarked, "collecting data for these simple measures is difficult; what would we do if we had to do something hard?" The data collection itself wasn't difficult though; instead, it was difficult to tailor the definitions and collection process to an existing process so that data could be collected unobtrusively and inexpensively, yet still add value to the process.
- **An organization needs commitment to persevere until measurement becomes a natural part of the overall software process.** Many times it would have been

easier to quit than to go on, especially within the SMPAT. But, the SMPAT chair stayed committed and that kept the program going. To date, the measurement program continues to make significant progress without becoming a natural part of the NAWC software process. Eventually, though, the measurement process could drive the shape of the software process.

- **Measurement must be tailored to a project's processes and needs.** Just like any process or standard, measurement must also be tailored to the processes used by a project as well as the needs of management for that project. The intent of the measurement program at NAWC has always been to augment the existing processes. To augment a process, the measures needed to be tailored to that process. Tailoring, though, requires that the measurement process be well defined because, at some point in time, NAWC intends to use the data collected and the trends that materialize as a historical perspective.
- **Ideally, data collection should be centralized in an organization and independent of the development projects.** The practice of having a group independent of the development projects and centralized in the organization was tested and proved to be very useful. A centralized group can grow its experience base and accomplish more than a set of unconnected groups. They can accomplish more by using their experience and tools as leverage to expand across many projects. Having the group independent of the government's program management group seemed to reduce some of the resistance of the contractor. In short, the quality improved as the metrics office serviced more projects and its processes became repeatable. As the process became repeatable, less time and resources were needed to service projects, thereby driving the overall cost for software measurement down.
- **Nonattribution of data and analysis results outside the project was key to participation and project buy-in.** Another big help in reducing the resistance between the contractor and the government's program management was ensuring the complete confidentiality and nonattribution of the data collected. The metrics office made the promise that data collected and the analysis results of that data for a specific project would be reported only to the project sponsor. The metrics office did reserve the right to use the data for examples, presentations, and comparisons in a historical manner, but only on a nonattribution basis. In fact, if there was a chance of a project being identified, the metrics office would not use the data. This pledge helped get projects to buy-in and participate and even motivated some to cooperate more than they may have without the pledge. It seems that nearly everyone we spoke with wanted to have the metrics office's help, but there always seemed to be concerns of retribution or abuses of the data once it left the control of the project or contractor. Pledging nonattribution and following through on that pledge alleviated those concerns.

- **It was difficult to implement measurement without "hooks" in the contract.** One item that led to the success of the pilot effort, and that everyone agreed would be important in using software measurement on any contract, was the need for "hooks" in the development process. For example, in increment one measures, this meant "hooks" such as monthly reporting of staff hours for all contractor staff and subcontractor staff, availability and regular updating of unit folders, etc. It also helped to have some "hooks" in the contract SOW for software measurement.
- **Measurement does not have to be expensive.** The software measurement program at NAWC was developed and implemented on a "shoestring" budget. Yet, in spite of the financial limitations, it was developed, implemented, and added value to the process. Now, the organization has a high-quality product and service it can market and sell to other organizations for a relatively inexpensive price and be self-supporting. The NAWC software measurement program was a fine example of how organizations can invest in themselves and add value to the organization.

7. Future Directions and Conclusions

To date, the SMPAT has developed a set of measures and an accompanying software measurement process and has tested the software measurement process on a pilot program. NAWC has also started a software measurement office and has developed a prototype measurement database. As of the summer of 1993, future plans and activities for the measurement program included:

- *Continue work on the analysis process:* Despite having completed the planning stage and actually starting to collect measurement data, the benefits from software measurement derive not from the act of defining and collecting measurement data, but from the information about the project that becomes available after analysis of the data.
- *Continue evolving increment one:* Two other factors indicate an evolving process: (1) as software technology matures, so will measurement technology; and (2) as the NAWC software improvement process effort progresses, the ensuing stability of its software process will enable the measurement initiative to include more sophisticated measures.
- *Develop training:* As the software measurement processes in increment one become more and more popular among NAWC projects, training will be needed. Initially, training will be needed to teach projects how to get measurement on the contract, how to collect data, and how to use the information provided by the metrics office.
- *Implement increment two (then three, and then four):* Software measurement is an ongoing evolutionary part of software engineering. The SMPAT has outlined four development increments for NAWC software measurement. Each increment is increasingly complex and difficult to implement. So far, the SMPAT has concentrated only on the first increment. The obvious next step is to have satisfactory results from the pilot on the first increment and then begin work on the remaining measures.

  The initial process that developed the processes for increment one needs to be repeated for the remaining increments. For each of the additional measures developed, detailed definitions, methods and forms for collecting the data, and the required analysis to make them useful also need to be developed. New increments will also affect the analysis of the previous increments as the analyses are merged and provide more global information.
- *Develop contracting tools:* To ensure the availability of measurement data, NAWC should have standardized language in its contracting documents and tools such as the SOW and CDRL. Included in these documents would be language to further allow the collection of software measurement data.

Conclusions

If you don't measure where you are now, and can't measure where you are going, it is impossible to tell if the changes made in the name of process improvement are actually improving the process or deteriorating it. For these reasons, a software measurement project should be an integral part of any software process improvement initiative.

Though the list of measures described in this report might seem simple, and possibly even trivial to some, there is a significant effort involved to define each measure in an unambiguous way and still be practical, to collect data consistently, and finally, to translate the data into useful information. It only seems easy and trivial, until you have to do it!

Bibliography

[2167A] Military Standard - Defense System Software Development, DOD-STD-2167A, February 1988.

[BASILI] Basili, V. and H. D. Rombach. "The TAME Project: Towards Improvement-Oriented Software Environment." *IEEE Transactions on Software Engineering*, (June 1988).

[BAUMERT] Baumert, J. H. and M. S. McWhinney. *Software Measures and the Capability Maturity Model* (CMU/SEI-92-TR-25). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, September 1992.

[BOEHM] Boehm, B., "Understanding and Controlling Software Costs." *IEEE Transactions on Software Engineering,* (October 1988)

[GOETHERT] Goethert, W. B., E. K. Bailey, and M. B. Busby. *Software Effort and Schedule Measurement: A Framework for Counting Staff Hours and Reporting Schedule Information* (CMU/SEI-92-TR-21 ADA 258279). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University , September 1992.

[HUMPHREY] Humphrev, W. S. *Managing the Software Process.* Reading, MA: Addison-Wesley Publishing Co., 1989.

[JONES] Jones, C. T. *Applied Software Measurement-Assuring Productivity and Quality.* New York, NY: McGraw-Hill, Inc., 1991.

[MOORE] Moore, R. L. *Methodology for the Management of Software Acquisition*, U.S. Army Missile Command, Redstone Arsenal, Alabama, June 1991.

[[NAVAIR] Naval Air Systems Command. *Avionics Software Metrics.* Department of the Navy, AVION Instruction 5235.1, June 1992.

[PARK] Park, R. E. *Software [illegible]e Measurement, A Framework for Counting Source Statements* (CMU/SEI-92-TR-20 ADA 258304). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, September 1992.

[PAULK] Paulk, M. C., B. Curtis, and M. B. Chrissis. *Capability Maturity Model for Software* (CMU/SEI-91-TR-24). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, September 1991.

[ROZUM92a] Rozum, J. A. *Software Measurement Concepts for Acquisition Program Managers* (CMU/SEI-92-TR-11 ADA 254177). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, June 1992.

[ROZUM92b] Rozum, James A. *NAWC Software Measurement Guide.* Software Engineering Institute, (SEI/NAWC-92-SR-1), October 1992 (available from NAWC-AD Warminster).

[ROZUM93] Rozum, J. A. and C. F. Koch. "NAWC-AD Software Measurement Program," *Proceedings of the Fifth Annual Software Technology Conference*, Salt Lake City, Utah: April 1993.

Appendix A — Acronyms

ASW	Anti-submarine warfare
CDR	Critical design review
CDRL	Contract data requirements list
CM	Configuration management
CMM	Capability Maturity Model
COTS	Commercial off the shelf
CSC	Computer software component
CSCI	Computer software configuration item
CSU	Computer software unit
CY	Calendar year
DoD	Department of Defense
FCA	Functional configuration audit
FQT	Functional qualification test
GFE	Government furnished equipment
GFI	Government furnished information
IDD	Interface design document
IRS	Interface requirements specification
ITG	Integration testing
IV&V	Independent verification & validation
LWL	Lower warning limit
NAVAIR	Naval Air System Command
NAWC	Naval Air Warfare Center
NAWCADWAR	NAWC Aircraft Division - Warminster
PCA	Physical configuration audit
PDW	Post deployment work
PDR	Preliminary design review
PFQT	Preliminary functional qualification test
PITG	Pre-integration testing
OT&E	Operational test & evaluation
RFP	Request for proposal
SCE	Software capability evaluation
SCP	Software change proposal
SDD	Software design document
SDF	Software development file (or folder)
SDP	Software development plan
SEI	Software Engineering Institute
SEPG	Software engineering process group
SLOC	Source lines of code
SPR	Software problem report
SMPAT	Software measurement process action team
SOW	Statement of work

SQA	Software quality assurance
SR	Special report
SRS	Software requirements specification
SSEB	Source selection evaluation board
SSR	Software specification review
STD	Software test description
STP	Software test plan
STR	Software test report
TBD	To be done (or designed or developed)
TR	Technical report
TRR	Test readiness review
TO&P	Technical objectives & plan
UWL	Upper warning limit
WBS	Work breakdown structure

Appendix B — Data Definition Forms

Data definition form #1: This form is used to define labor categories that are then used in the staff and effort measures.

Data definition form #2: This set of forms is used to define what effort is included and excluded in the staff-hour measures. In addition, it identifies what staff-hour measures are to be collected.

Data definition form #3: This set of forms defines what is included and excluded in counting source lines of code.

NAWCADWAR Data Definition Form #1

Organization		Project		
Delivery #/Build #				
Labor Category		**Experience (years)**		
A =	Program Manager	more than		years
B =	Technical Manager	more than		years
C =	Sr. Systems Engineer	more than		years
D =	Systems Engineer	more than		years
E =	Jr. Systems Engineer	more than		years
F =	Sr. Software Engineer	more than		years
G =	Software Engineer	more than		years
H =	Jr. Software Engineer	more than		years
I =	Doc. Spec./Tech Writer	more than		years
___		more than		years
		more than		years
		more than		years

NAWCADWAR Data Definition Form #2 (page 1 of 3)

Definition Name:	NAWC Aircraft Division - Warminster
Date:	10/1/92
Originator:	SMPAT
Page:	1 of 3

	Totals include	Totals exclude	Report totals
Type of Labor			
Direct	✓		
Indirect		✓	
Hour Information			
Regular time			✓
Salaried	✓		
Hourly	✓		
Overtime			✓
Salaried			
Compensated (paid)	✓		
Uncompensated (unpaid)	✓		
Hourly			
Compensated (paid)	✓		
Uncompensated (unpaid)	✓		
Employment Class			
Reporting organization			
Full time	✓		
Part time	✓		
Contract			
Temporary employees	✓		
Subcontractor working on task with reporting organization	✓		
Subcontractor working on subcontracted task		✓	✓
Consultants		✓	✓
Labor Class			
Software management			
Level 1	✓		
Level 2	✓		
Level 3		✓	
Higher		✓	
Technical analysts & designers			
System engineer	✓		
Software engineer/analyst	✓		
Programmer	✓		
Test personnel			
CSCI-to-CSCI integration	✓		
IV&V	✓		
Test & evaluation group (HW-SW)	✓		
Software quality assurance	✓		
Software configuration management	✓		
Program librarian		✓	
Database administrator		✓	
Documentation/publications	✓		
Training personnel	✓		
Support staff		✓	

NAWCADWAR Data Definition Form #2 (page 2 of 3)

Definition Name: **NAWC Aircraft Division - Warminster** Page: 2 of 3

	Totals include	Totals exclude	Report totals
Activity	▨	▨	▨
Development	▨	▨	✓
Primary development activity	✓		
Development support activities	▨	▨	
Concept demo/prototypes	✓		
Tools development, acquisition, installation, & support	✓		
Non-delivered software & test drivers	✓		
Maintenance	▨	▨	✓
Repair	✓		
Enhancements/major updates	✓		
Product-Level Functions	▨	▨	▨
CSCI-Level Functions (Major Functional Element)	▨	▨	
Software requirements analysis	✓		
Design	▨	▨	
Preliminary design	✓		
Detailed design	✓		
Code & development testing	▨	▨	
Code & unit testing	✓		
Function (CSC) integration and testing	✓		
CSCI integration & testing	✓		
IV&V		✓	
Management	✓		
Software quality assurance	✓		
Configuration management	✓		
Documentation	✓		
Rework	▨	▨	✓
Software .equirements	✓		
Software implementation	▨	▨	
Re-design	✓		
Re-coding	✓		
Re-testing	✓		
Documentation	✓		
Build-Level Functions (Customer Release)	▨	▨	▨
CSCI-to-CSCI integration & checkout	✓		
Hardware/software integration and test	✓		
Management	✓		
Software quality assurance	✓		
Configuration management	✓		
Documentation	✓		
IV&V		✓	

NAWCADWAR Data Definition Form #2 (page 3 of 3)

Definition Name: **NAWC Aircraft Division - Warminster** Page: 3 of 3

Product-Level Functions (continued)	Totals include	Totals exclude	Report totals
System-Level Functions	▨	▨	✓
System requirements & design	▨	▨	
System requirements analysis	✓		
System design	✓		
Software requirements analysis	✓		✓
Integration, test, & evaluation	▨	▨	✓
System integration & testing	✓		
Testing & evaluation	✓		
Production and deployment	✓		
Management	✓		
Software quality assurance	✓		
Configuration management	✓		
Data	✓		
Training	▨	▨	
Training of development employees	✓		
Customer training	✓		
Support		✓	

NAWCADWAR Data Definition Form #3 (page 1 of 3)

Measurement unit:	**Physical source lines**	☐
	Logical source statements	✓

Statement type Definition ✓ Data array ☐		**Includes**	**Excludes**
When a line or statement contains more than one type, classify it as the type with the highest precedence.		▨	▨
1 Executable **Order of precedence ->**	1	✓	
2 Nonexecutable	▨	▨	▨
3 Declarations	2	✓	
4 Compiler directives	3	✓	
5 Comments	▨	▨	▨
6 On their own lines	4		✓
7 On lines with source code	5		✓
8 Banners and nonblank spacers	6		✓
9 Blank (empty) comments	7		✓
10 Blank lines	8		✓
11 Format Statements	9	✓	
12 Data Statements	10	✓	

How produced Definition ✓ Data array ☐	**Includes**	**Excludes**
1 Programmed	✓	
2 Generated with source code generators	✓	
3 Converted with automated translators	✓	
4 Copied or reused without change	✓	
5 Modified	✓	
6 Removed		✓
7		
8		

Origin Definition ✓ Data array ☐	**Includes**	**Excludes**
1 New work: no prior existence	✓	
2 Prior work: taken or adapted from	▨	▨
3 A previous version, build, or release	✓	
4 Commercial, off-the-shelf software (COTS)	✓	
5 Government furnished software (GFS)	✓	
6 Another product	✓	
7 A vendor-supplied language support library (unmodified)	✓	
8 A vendor-supplied operating system (e.g., UNIX) (unmodified)	✓	
9 A modified or local language support library or operating system	✓	
10 A commercial library	✓	
11 A reuse library (software designed for reuse)	✓	
12 Another software component or library	✓	
13		
14		

Usage Definition ✓ Data array ☐	**Includes**	**Excludes**
1 In or as part of the primary product	✓	
2 External to or in support of the primary product		✓
3		

NAWCADWAR Data Definition Form #3 (page 2 of 3)

Delivery Definition [✓] Data array [ ]	Includes	Excludes
1 Delivered	▨	▨
2 Delivered as source	✓	
3 Delivered in compiled or executable form, but not as source	✓	
4 Not delivered	▨	▨
5 Under configuration control		✓
6 Not under configuration control		✓
7		
8		

Functionality Definition [ ] Data array [ ]	Includes	Excludes
1 Operative		
2 Inoperative (dead, bypassed, unused, unreferenced, or unaccessed)		
3		
4		

Replications Definition [ ] Data array [ ]	Includes	Excludes
1 Master source statements (originals)		
2 Copies of master statements physically repeated in the master code		
3 Copies inserted, instantiated, or expanded when compiling or linking		
4 Postproduction replicates—as in distributed, redundant, or reparameterized systems		
5		
6		

Development status Definition [✓] Data array [ ]	Includes	Excludes
1 Designed		✓
2 Coded		✓
3 Unit tests completed and unit under configuration control	✓	
4 Integrated into components		
5 Test readiness review completed		
6 Software (CSCI) tests completed		
7 System tests completed		
8		
9		
10		
11		

NAWCADWAR Data Definition Form #3 (page 3 of 3)

		Definition Includes	Definition Excludes
Clarifications (general)		▨	▨
1 Nulls, continues, and no-ops **Are assigned to type ->**			✓
2 Empty statements (e.g., ;; and lone semicolons on separate lines)			✓
3 Statements that instantiate generics	1	✓	
4 Begin...end and {...} pairs used as executable statements	1	✓	
5 Begin...end and {...} pairs that delimit (sub)program bodies	1	✓	
6 Logical expressions used as test conditions	1	✓	
7 Expression evaluations used as subprogram arguments	1	✓	
8 End symbols that terminate executable statements			✓
9 End symbols that terminate declarations or (sub)program bodies			✓
10 Then, else, and otherwise symbols	1	✓	
11 Elseif statements	1	✓	
12 Keywords like procedure division, interface, implementation, etc.	2	✓	
13 Format Statements	9	✓	
14 Print Statements	1	✓	
15 Data Statements	10	✓	
16 Compiler Directives	1	✓	
17			
Clarifications (language specific)			
Ada		▨	▨
1 Null statements			✓
2 Statements that instantiate generics	1	✓	
3 End symbols that terminate declarations or (sub)program bodies			✓
4 Block statements (e.g., begin...end)	1	✓	
5			
6			
Assembly		▨	▨
1 Macro calls	1	✓	
2 Macro expansions			✓
3			
4			
FORTRAN		▨	▨
1 END statements			✓
2 Format statements	9	✓	
3 Entry statements	2	✓	
4			
5			
6			
C and C++		▨	▨
1 Null statement (e.g., ";" by itself to indicate an empty body)			✓
2 Expression statements (expressions terminated by semicolons)	1	✓	
3 Expressions separated by semicolons, as in a "for" statement	1	✓	
4 Block statements (e.g., {...} with no terminating semicolon)	1	✓	
5 "{", "}", or "};" on line by itself when part of a declaration			✓
6 "{" or "}" on line by itself when part of an executable statement			✓
7			

Appendix C — Data Collection Forms

Data collection form #1: In the absence of electronic data reporting, this form is used to collect data on the number of full-time, part-time, and total staff for a project. The data are collected by labor class category.

Data collection form #2: In the absence of electronic data reporting, this form is used to collect data on the number of staff-hours by labor category for each functional area.

Data collection form #3: This form is used to collect data on the planned and actual completion dates of identified milestones and deliverables.

Data collection form #4: This form is used to collect data on the number of work items completed and planned to be completed for identified time periods.

Data collection from #5: This form is used to collect data on the actual number of source lines of code developed, modified, and reused as well as the estimated number of lines of code remaining for each area by language.

Data collection form #6: This set of forms is used to collect data on the number of software problem reports along various, identified characteristics.

NAWCADWAR Data Collection Form #1

Organization	Project
Delivery #/Build #	Date of Data Collection

Labor Class	Full-Time Staff		Part-Time Staff		Staff	
	Planned	Actual	Planned	Actual	Added	Lost
A						
B						
C						
D						
E						
F						
G						
H						
I						

NAWCADWAR Data Collection Form #2

Organization	Project
Delivery #/Build #	Date of Data Collection

Functional Areas	**Staff Hours by Labor Category**								
	A	B	C	D	E	F	G	H	I
System / Task Planning									
System Integration and Test									
Software Task Planning									
Software Reqt. Anal. / Sys. Des.									
Software High Level Design									
Software Detailed Design									
Software Code and Unit Test									
Software Integration & Testing									
Software Quality Assurance									
Software Configuration Mgt.									
Program Management									
Test Support									
Training									
Tech Data / Tech Manuals									
TOTALS									

NAWCADWAR Data Collection Form #3

Organization	Project	
Delivery #/Build #	Date of Data Collection	
Milestones	Planned	Actual
Software Specification Review (SSR)		
Preliminary Design Review (PDR)		
Critical Design Review (CDR)		
Test Readiness Review (TRR)		
Formal Qualification Testing (FQT)		
Functional Configuration Audit (FCA)		
Physical Configuration Audit (PCA)		
Deliverables	Planned	Actual
Software Requirements Spec. (SRS)		
Interface Requirements Spec. (IRS)		
Software Design Document (SDD)		
Interface Design Document (IDD)		
Software Test Plan (STP)		
Software Test Description (STD)		
Software Test Report (SPR)		
Software Development Plan (SDP)		

NAWCADWAR Data Collection Form #4

Organization	Project
Delivery #/Build #	Date of Data Collection

Work Item	Previously Completed	Completed this Period	Planned this period	Total Completed	Total Planned
Requirements in SRS	N/A	N/A	N/A		
SRS Reqts. in SDD	N/A	N/A	N/A		
SRS Reqts. in IDD	N/A	N/A	N/A		
CSUs Designed					
CSUs Coded / Tested					
Regression Tests					
Integration Tests					
FQT Tests					
Reqts. Tested in FQT					

NAWCADWAR Data Collection Form #5

Organization	Project
Delivery #/Build #	Date of Data Collection
CSCI ID	# of CSUs

		SLOC		
		New	Modified	Reused
Language	Estimated			
____________	Actual			
	Total			
Language	Estimated			
____________	Actual			
	Total			
Language	Estimated			
____________	Actual			
	Total			

NAWCADWAR Data Collection Form #6 (page 1 of 3)

Organization	Project
Delivery #/Build #	Date of Data Collection

	SPRs	
	This Report	To Date
Received		
Open		
Closed		

	Number of days SPRs are open				
	< 30	30 to 60	61 to 90	> 90	Total
Priority 1					
Priority 2					
Priority 3					
Priority 4					
Total					
SCPs					

NAWCADWAR Data Collection Form #6 (page 2 of 3)

Organization:

Project:

Date of Data Collection:

SPRs Opened This Reporting Period (By Priority for Each CSCI)

CSCI	1	2	3	4	Total
TOTALS					

NAWCADWAR Data Collection Form #6 (page 3 of 3)

Organization
Project
Date of Data Collection

Test Source	SPRs Opened This Reporting Period (By Priority)					SPRs Closed This Reporting Period (By Priority)				
	1	2	3	4	Total	1	2	3	4	Total
PITG										
ITG										
PFQT										
FQT										
IV&V										
Beta										
OT&E										
Field										
TOTALS										

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-93-TR-07	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESC-TR-93-184
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute / 6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office
6c. ADDRESS (city, state, and zip code) Carnegie Mellon University Pittsburgh PA 15213	7b. ADDRESS (city, state, and zip code) HQ ESC/ENS 5 Eglin Street Hanscom AFB, MA 01731-2116
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office / 8b. OFFICE SYMBOL (if applicable) ESC/ENS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003
8c. ADDRESS (city, state, and zip code)) Carnegie Mellon University Pittsburgh PA 15213	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO 63756E; PROJECT NO. N/A; TASK NO N/A; WORK UNIT NO. N/A

11. TITLE (Include Security Classification)
The SEI and NAWC: Working Together to Establish a Software Measurement Program

12. PERSONAL AUTHOR(S)
James A. Rozum

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (year, month, day)	15. PAGE COUNT
Final	FROM TO	December 1993	95 pp.

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (continue on reverse of necessary and identify by block number)
FIELD	GROUP	SUB. GR.	process action team
			project management
			software measurement
			software process improvement

19. ABSTRACT (continue on reverse if necessary and identify by block number)

In 1990, the Software Engineering Institute (SEI) and the then Naval Air Development Center (NADC) in Warminster, Pennsylvania (now Naval Air Warfare Center, Aircraft Division, Warminster) signed an agreement to jointly develop a software measurement program for NADC - Warminster. To help with that development, a software measurement process action team (SMPAT) was formed with members from the SEI and NADC. The SMPAT's responsibility was to plan, develop, and assist in the implementation of the measurement program. The purpose of this technical report is to document and make available to the software community the process and methods used, experience gained, and some lessons learned in establishing the software measurement program at NADC.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ■ SAME AS RPT ☐ DTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution	
22a. NAME OF RESPONSIBLE INDIVIDUAL Thomas R. Miller, Lt Col, USAF	22b. TELEPHONE NUMBER (include area code) (412) 268-7631	22c. OFFICE SYMBOL ESC/ENS (SEI)

ABSTRACT — continued from page one, block 19

This report is meant to help organizations that desire to start a software measurement program or have been struggling with such a program by providing an example of one organization that has also struggled to establish a software measurement program. To help an organization, real-life examples of how software measures were defined, collected, and used to improve the management process are included.