PART EIGHT

Project Measurement Case Studies

Practical Software and Systems Measurement: A Foundation for Objective Project Management was developed to show how measurement can be used to address the issues faced by today's government and industry project managers. To better illustrate how the measurement process is implemented for different types of projects, this part of the Guide contains three case studies. These case studies describe how the measurement process is tailored and applied to meet specific project management requirements.

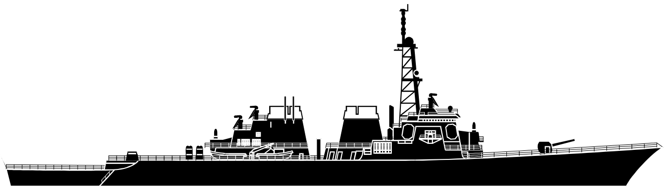
The PSM case studies address the issues and challenges that most project managers face in planning, implementing, and maintaining software and systems projects. The case studies concentrate on the issues that the project manager must address with respect to managing the project within defined acquisition and technical constraints. They show how measurement is used to help make decisions concerning project issues.

Although the Practical Software and Systems Measurement case study parameters are based on actual project characteristics, the project scenarios, including the described system architectures, project names, and project organizations are fictitious.

The case studies are organized into three sub-sections:

- **Part A, Weapons System Case Study,** is based upon a major shipboard Weapons development.
- **Part B, Information System Case Study**, describes an information system used to manage military personnel information.
- **Part C, Operations and Maintenance Case Study**, is based on a deployed radar system in maintenance.

[This page intentionally left blank.]



PART 8A

Weapons System Case Study

The Weapons System Case Study describes the development of a complex shipboard weapons system designed to integrate multiple-platform target engagement and weapons-management functions into an existing system baseline. In this scenario, measurement is used to help estimate, plan, and track the software development from the inception of the program through system deployment. The development approach is based on the upgrade of an existing system using commercial-off-the-shelf (COTS) components and reused software in a revised architecture. The supplier is a competitively selected contractor who works closely with the Navy Program Manager to identify and resolve typical issues in a large development program. These issues include software requirements and size growth, incremental schedule slips, and overall software development productivity shortfalls.

The Weapons System Case Study is organized into three chapters:

- **Chapter 1, Program Overview,** describes the technical and management aspects of the software development effort.

- **Chapter 2, Program Planning and Acquisition,** shows how measurement can be used to estimate the basic size, effort and schedule parameters and to define and evaluate a realistic project management plan.

- **Chapter 3, Software Development,** illustrates how measurement helps to identify and track software issues, and how the program manager uses measurement information to evaluate development status and make informed decisions.

1

Program Overview

This chapter introduces the example Navy program and describes the technical and management aspects of the development effort. The program scenario is based on a major program upgrade to an existing Navy surface ship Command, Control, Communications, Computer, and Intelligence (C4I) system. The upgrade integrates multiple-platform target engagement and weapons management functions into an existing software functional baseline. It includes the addition of new software functions to the system, as well as modifications to the existing software baseline.

1.1 Introduction

In the early 1990's, the Navy began to recognize a growing need for its ships and aircraft to operate interactively in a multiple-threat environment. This need was clearly demonstrated during the Gulf War, when well-coordinated engagements, which integrated the capabilities of a number of different platforms, provided significant tactical advantages.

To define its changing mission requirements, the Navy initiated a concept study to determine the feasibility and effectiveness of integrating a multiple-platform target engagement capability into the fleet. The results of the study, completed in 1994, validated the need for the proposed engagement capabilities and recommended an implementation approach that built upon the Navy's existing C^4I tactical systems on various platforms. The study recommended that the Navy initially focus on the upgrade of its existing surface combatants with new communications, engagement management, and weapons control functions. These new functions would be designed to allow two or more ships to engage the enemy as a single entity. With the new capabilities, one ship would be able to manage the overall sensor and target scenarios for the entire group and assign, launch, and control the weapons on the other ships using advanced tactical communications links.

The Navy decided that the Arleigh Burke DDG 51 class of guided missile destroyers (DDG) would be the first ships to receive the capability upgrade, as it was the largest and most modern class of DDGs in the fleet. It named the program the **DDG 51 Surface Ship Concurrent Weapons Engagement Upgrade Program**, or **DDG 51 SCWE** for short. The objective of the DDG 51 SCWE program was to define, develop and integrate a new concurrent weapons engagement function into the existing C^4I system on the Arleigh Burke DDGs. Most of the efforts were to be focused on the coordinated employment of long-range, surface-launched weapons, with an emphasis on the Tomahawk Cruise Missile.

The DDG 51 SCWE program was projected to require significant changes in the architecture of the existing DDG 51 C^4I system, especially with respect to the software. Existing software functions and interfaces required numerous changes, and the multiple platform communications, target management, and weapons management functions had to be developed and integrated. New acquisition policies made the use of an open systems architecture (OSA) and commercial-off-the-shelf (COTS) software components mandatory. The overall business environment required that the program be well managed in terms of delivered functionality and in meeting pre-defined cost and schedule objectives.

The Navy recognized the critical nature of the software development component of the DDG 51 Surface Ship Concurrent Weapons Engagement Upgrade Program and emphasized the need for effective software management as part of the overall program management approach. Understanding this need, the Naval Sea

Systems Command (NAVSEA) assigned Captain Katherine McLain, USN, as the Program Manager. Captain McLain held an advanced degree in Electrical Engineering from Stanford University, and she had served as the technical manager on several previously successful Navy development programs. Her last assignment was Deputy Program Manager for an upgrade to the command and control system for the carrier fleet. After completing the Program Manager's course at the Defense Systems Management College (DSMC), Captain McLain assembled her program management team at NAVSEA. Her office was designated as PMO-551. The award date for DDG 51 SCWE Engineering and Manufacturing Development (EMD) was projected for mid-1996. To ensure a successful program, a considerable amount of work had to be completed before award.

1.2 Program Technical Approach

1.2.1 System Requirements Definition and Design Analysis

Based on her previous experience, Captain McLain was familiar with the software architecture and capabilities of the existing DDG 51 C^4I system. Like most of the large Navy systems developed in the late 1980's, the system on the Arleigh Burke DDG class was built around the AN/UYK-43 Navy standard computer, which centrally handled the processing for most of the system's different functions. The original C^4I systems on the DDG 51's had been incrementally upgraded since they were first deployed to integrate new sensor and weapons capabilities. Over time, the system design had proven to be effective and reliable.

The software for the DDG 51 C^4I system was implemented largely in CMS-2, the Navy's pre-Ada standard high order programming language. The functions where real-time processing and timing considerations were critical were coded in assembly language. The original software had been developed using a modified DoD-STD-2167 software development process and was currently being maintained by the original supplier under a separate maintenance contract.

The mission requirements driving the DDG 51 SCWE program provided some significant technical and program management challenges for PMO-551. Captain McLain felt that one of the keys to a successful development program was a well-defined set of system requirements. As part of the acquisition strategy, PMO-551 awarded a series of competitive System Requirements Definition-Design Analysis Study Contracts. These design study contracts were specifically implemented to accomplish the following:

- Provide a definitive analysis and characterization of the existing DDG 51 C4I system hardware and software architectures
- Develop an approved set of system-level requirements for inclusion in the EMD Request For Proposal (RFP)
- Develop innovative system design alternatives. These alternatives in particular were focused on the use of COTS hardware and software components and on the integration of an OSA into the existing system to support future capability growth

1.2.2 DDG 51 C^4I Baseline System Description

The results of the System Requirements Definition-Design Analysis Study efforts provided a detailed characterization of the existing DDG 51 C^4I software architecture. Figure 8A-1, a simplified system diagram, shows that the system consisted of six primary software functions, all resident in the AN/UYK-43 computer. Data interfaces to the External Communications subsystems, the Weapons subsystems, and to own-ship sensors encompassing Navigation, Radar, Sonar, and Electronic Support Measures (ESM), were through the System Control software function using a Navy Tactical Data System (NTDS) interface protocol. Two-way data communications to the Command Display and Control consoles was also provided by the System Control software through an NTDS interface.

Each of the six primary software functions in the system was comprised of three to six software Configuration Items (CIs), as defined in DoD-STD-2167. In all, there were 24 CIs in the baseline system. The software architecture was well defined, and the original supplier had done an excellent job of allocating and mapping the original software requirements to the CIs. There was a full set of technical specifications available, but they had not been kept uniformly up to date, especially with respect to the incremental design changes.

The DDG 51 C^{4}I system software was relatively large and somewhat complex. The various software functions worked together to integrate real-time data from a variety of distinct combat and ship control subsystems and processed the data into the information needed to effectively engage enemy targets.

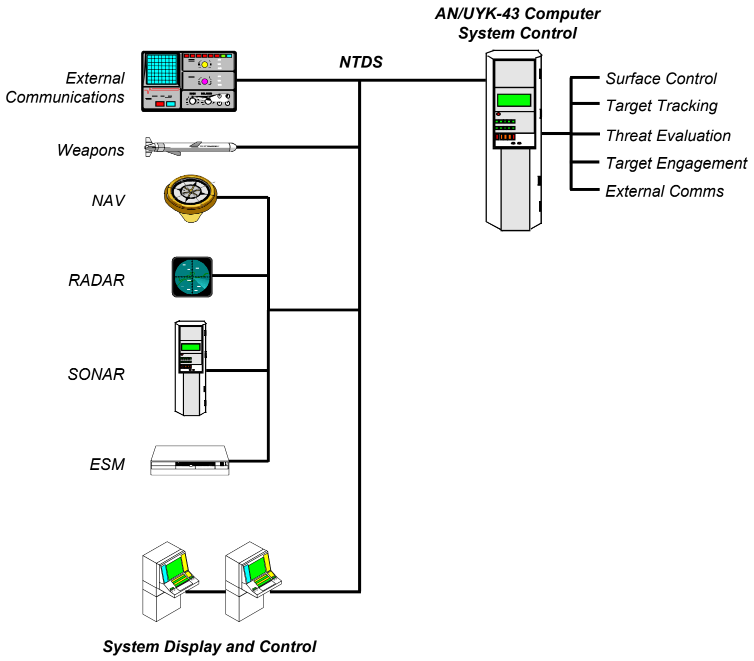


Figure 8A-1. The DDG 51 Weapons System Software Architecture

Each of the six primary software functions addressed a unique set of functional requirements:

- **System Control** - The System Control function included the AN/UYK-43 operating system and provided the primary software services functions for the system. Its functions included system database management, initial program load, configuration and reconfiguration management, and display control.

- **Surface Control** - The Surface Control function addressed own-ship maneuvering and navigation requirements and calculated ship's heading, speed, and position on a real-time basis. It also included capabilities that helped position the ship with respect to other surface contacts.

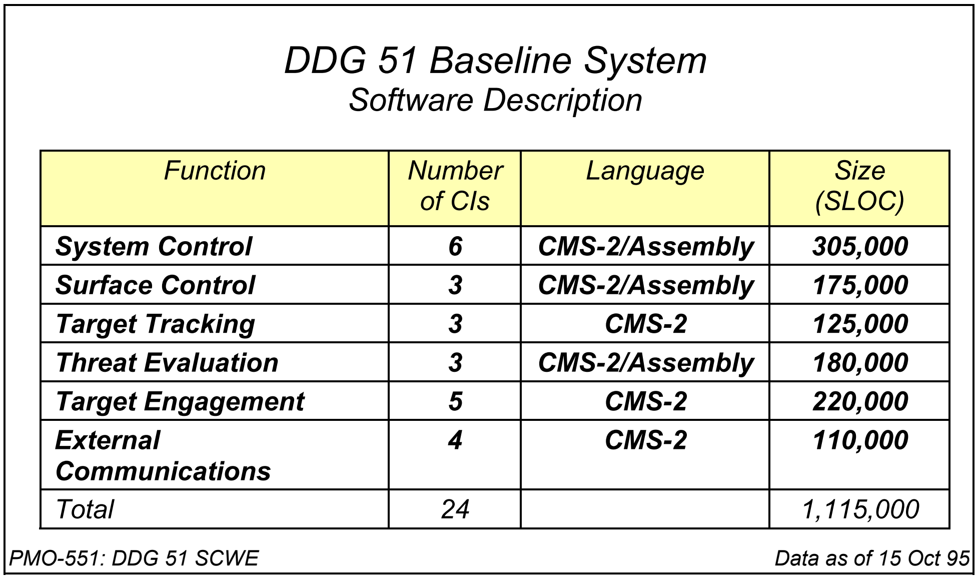
- **Target Tracking** - The Target Tracking function integrated and correlated all sensor data, and calculated, evaluated, and tracked surface, subsurface, and air contacts on a real-time basis.

- **Threat Evaluation** - The Threat Evaluation function correlated all of the sensor data from all targets and, through a series of complex threat algorithms, calculated and prioritized each target within an overall threat profile.

- **Target Engagement** - The Target Engagement function included software that managed the overall enemy engagement and controlled all weapons allocations to individual targets. It also assigned weapons presets based on the calculated target parameters. This function was one of the most critical in the system.

- **External Communications** - The External Communications function provided interfaces between the C4I system and a number of tactical digital communications data links. These data links provided for the exchange of contact and targeting information with other off-ship platforms.

Together, the DDG 51 C^4I system software functions included over one million logical lines of source code distributed among 24 CIs as shown in Figure 8A-2.



DDG 51 Baseline System
Software Description

Function	*Number of CIs*	*Language*	*Size (SLOC)*
System Control	***6***	***CMS-2/Assembly***	***305,000***
Surface Control	***3***	***CMS-2/Assembly***	***175,000***
Target Tracking	***3***	***CMS-2***	***125,000***
Threat Evaluation	***3***	***CMS-2/Assembly***	***180,000***
Target Engagement	***5***	***CMS-2***	***220,000***
External Communications	***4***	***CMS-2***	***110,000***
Total	*24*		*1,115,000*

PMO-551: DDG 51 SCWE *Data as of 15 Oct 95*

Figure 8A-2. System Software Functions

1.2.3 System Requirements and Design Recommendations

The System Requirements Definition-Design Analysis study effort provided a definitive set of system level requirements for the DDG 51 SCWE upgrade program. After reviewing the requirements with her staff, Captain McLain had a clear understanding of the magnitude of the changes required for the existing DDG 51 C^4I system. She knew that the new multi-ship engagement functions would have a significant impact on the existing system and software architectures. She also estimated that the current software baseline would more than double in size.

In addition to the new multiple-platform engagement management and weapons control functions, the system level requirements included the need for:

- New display processing capabilities

- New assignable command and display workstations

- Automatic reconfiguration of the engagement control functions in the event of a platform specific failure

- Enhanced weapons safety provisions

- Advanced multiple ship and aircraft contact correlation
- Additional secure digital data links
- An increase in the overall system processing capacity

Even at this point in the program, Captain McLain knew that managing the requirements, especially those allocated to the software, would be important to the success of the upgrade program.

Given the large amount of functionality that was to be added to the baseline DDG 51 system, the System Requirements Definition-Design Analysis studies also proposed a number of system and software design alternatives that addressed the Navy's desire for development affordability and life-cycle cost savings. These alternatives were all based upon retaining a large part of the baseline system hardware and software and adding the new capabilities using COTS components integrated via an OSA local area network. In all cases, the alternatives addressed the addition of new processing and display capabilities using advanced display workstations.

The design alternatives outlined in the study maintained much of the integrity of the existing system hardware and software. In addition, they addressed the Navy's policy to embrace open commercial interface standards and COTS products in implementing the new functionality.

1.3 Program Management Approach

With the system specifications and the design studies completed, Captain McLain began to concentrate on the program's acquisition requirements. With her own program office personnel, and support from the Naval Surface Warfare Center (NSWC) in Dahlgren, Virginia, Captain McLain believed she had a capable acquisition team, especially with respect to software.

With the changes in the DoD business environment over the past several years, Captain McLain knew that the DDG 51 SCWE program would be visible within the Navy and DoD. It was one of the first major programs to fully address the DoD's acquisition reform requirements, which included Cost As An Independent Variable (CAIV), the extensive use of commercial standards, COTS hardware and software, software reuse, and the integration of an open system architecture.

One of the key aspects of acquisition reform was its emphasis on less supplier oversight by the acquisition organization. Captain McLain's program office, PMO-551, was assigned half the number of personnel than would have been typical only a few years earlier. She knew that her office would have to function much more efficiently than it had in the past. This requirement led to several critical software decisions by Captain McLain:

- The supplier had to have a mature software development process, and the supplier's overall capability with respect to life-cycle process would be a key consideration in source selection.
- Insight into the life-cycle processes and products, across all activities and life-cycle phases, would be provided by an issue-driven project measurement process. Both PMO-551 and the supplier would use project measurement to identify and manage the software development issues. When one of her senior staff members protested that they couldn't afford a measurement program, Captain McLain countered that they couldn't afford not to have one.
- The winning contractor would be required to implement a formal risk management program. The results of the risk assessment would be used as a major input in tailoring the measurement program. The measurements, in turn, would be used to help manage risks.

- The government and supplier organizations would function with software and systems engineering integrated product team (IPT). IPTs would also be established for weapons and for electronic countermeasures. The IPT would be given broad decision-making authority in addressing issues within the software arena. One of the IPT tasks was to integrate the measurement activities with risk management and financial performance management to establish an Earned Value reporting system. Decisions requiring coordination with other IPTs would be addressed by an overarching IPT, of which Captain McLain was the lead.

- Given the funding constraints within the current environment, the DDG 51 SCWE program would implement Cost As An Independent Variable (CAIV). Cost would be a major consideration in evaluating design alternatives. The IPTs would be given authority to consider a variety of design options as long as critical performance and reliability requirements were met. There was the possibility of tradeoffs between the hardware and software functions in order to meet the overall program cost objectives. Captain McLain would monitor software requirements growth and financial performance closely.

- The software would be developed using a tailored MIL-STD-498 development process. Along with this, a detailed software Work Breakdown Structure (WBS) would be implemented to manage the program's development products and activities.

Captain McLain planned to award the development contract to a capable software supplier with a proven performance history. She made it clear that she expected both her PMO-551 organization and the supplier to address the software issues in an objective manner. Captain McLain knew that delivering software that meets the specified requirements to the fleet within the program's schedule and funding constraints would be a significant challenge.

2

Program Planning and Acquisition

With the system requirements completed, PMO-551 began to focus on the detailed planning for the DDG 51 Surface Ship Concurrent Weapons Engagement Upgrade Program. Before awarding the development contract, Captain McLain and the Navy program team had to define a feasible project management plan, issue the Request for Proposal (RFP) and evaluate the submitted proposals during source selection. Even at this early planning stage, Captain McLain used information derived from the project measurement process to support her planning objectives.

This chapter of the case study shows how project measurement can help during the Program Planning phase of software development. ***The activities that take place during this phase set the stage for project success or failure.*** Their importance cannot be over-emphasized. It is during this time that the program manager implements the measurement process as an integral part of the overall program management structure. Project measurement is used to ensure that the software development estimates are realistic, that the plan is feasible, and that the software supplier has the capability to successfully complete the job.

2.1 Software Program Planning

The most important software planning task for Captain McLain and her staff was to develop a realistic DDG 51 SCWE software implementation plan. Aware of the direct relationships between the overall size of the software and development cost and schedule, Captain McLain and her engineering team generated estimates for the key parameters.

PMO-551 began with a preliminary allocation of the system requirements to a notional set of software components, keeping in mind that they would be retaining much of the existing software and using a significant amount of COTS software to implement the new functions. Based on these requirements, and the size of the existing code, the team estimated the size of the software to be developed. They knew that there was a great deal of uncertainty in those estimates, but this was a starting point for setting the basic size, cost, and schedule parameters for the program.

Using the size estimates, they generated estimates of development effort and schedule using two techniques. First, they had their own engineering rules-of-thumb for development productivity (lines of code per staff month) and code-production rates (lines of code per calendar month). These rules-of-thumb were derived from past experiences with similar C^4I programs. In both cases, these engineering estimates encompassed the key software development activities (software requirements analysis through system integration and test). Secondly, they used a commercially-available software cost estimating model. With this model, they could express the uncertainty of their size estimates by entering a range of estimates for each of the notional system components from very low to very high. The model then estimated a range of values for cost and schedule. At the low end, the cost and schedule were estimated to occur with a likelihood of only 5 percent; at the upper end, the likelihood was 95 percent; the middle value would occur with a likelihood of 50 percent.

From these estimates, Captain McLain concluded that the schedule required to realistically complete the software was between 64 and 78 months, starting with contract award and ending with hardware-software integration and test and delivery to the shipyard for the start of operational testing. Unfortunately, this time was somewhat longer than the schedule the Navy had defined. The ship deployment and shipyard availability schedules were driving the DDG 51 SCWE development schedule, and the software was the "long pole in the tent." Based on her analysis, Captain McLain knew that the schedule was going to be high risk, and took steps to address this issue in her plan.

Captain McLain understood that the program budget and functional requirements were essentially set; therefore, she looked at several options for reducing the planned software development schedule.

Captain McLain updated her plan to include the following:

- More parallel implementation of the software development activities. This included an incremental development approach, with 1) the functionality developed and integrated into multiple increments and, 2) the overlapping of specific software implementation, integration, and test activities.

- Maximized use of COTS and non-developed (NDI) software, and the reuse of as much of the existing code as possible

- Assumption of relatively high software development productivity, based on her plan to make the supplier's life-cycle process capability a key criterion for contract award

After these modifications, the PMO-551 re-ran the software estimates. Specifically, the PMO-551 planning team assumed the amount of code that had to be developed was smaller due to the use of additional COTS software and more reused software components from the baseline system. The resulting DDG 51 SCWE software development schedule showed that the full set of software requirements could be implemented in 66 months, within the original cost objective. This estimate was close to the delivery target date set by the Navy.

From a technical perspective, Captain McLain decided that any new software developed for the system should be implemented in Ada, using the Ada 95 standard.

Since the start of the program, Captain McLain had emphasized the value of a formal risk management process. Captain McLain used the risk assessment results to help identify the software issues that the measurement process should address. In reviewing the results of the risk assessment, the following software risks were identified:

- The software development schedule was, as expected, highly constrained. Based on a number of trial runs with a software cost model, the planning team assigned a probability of 0.80 that the delivery of the software to systems integration would be late. They assigned an impact of 10 to this event, since the software was on the critical path for system delivery and operational test. Captain McLain knew that the project schedule was an extremely high-priority issue. If the supplier could not meet the schedule, Captain McLain wanted to know as early as possible so she could consider alternative actions. The relative risk exposure, on a scale of 1 to 10, was assigned an 8 for the schedule issue. Further quantification of the impact in terms of dollars showed that any schedule slip would have a significant cost impact on the program.

- The risk assessment indicated a possibility that the DBMS access routines would not meet the real-time response requirements. A probability of 0.25 was assigned to this event, with a relative impact of 10. This event was assigned the highest relative impact because not meeting the real-time response requirements could result in loss of mission capability. The resulting risk exposure was 2.5.

- The functional architecture for the Electronic Support Measures was untried. There was a risk of interference between the receivers and transmitters. This interference, however, could be corrected by software. The risk analyst assigned a 0.30 probability to this event and an impact of 8. The impact was assigned a high priority because the mission capability could be seriously jeopardized. The risk exposure was 2.4.

- There was a possibility that the new surface missile might not be ready for deployment during the operational test and evaluation phase. The analyst assigned a probability of 0.20 and an impact of 2, since the primary mission could still be performed without the new missile. The risk exposure was 0.4. This was a program-level risk, rather than a software-level risk.

- The successful use of COTS and the incorporation of much of the existing software baseline was key to the whole strategy of completing the program within the schedule and cost constraints. If serious problems arose with the COTS software or with the baseline software, a major replan would be necessary. The analyst assigned a probably of .35 to this event and an impact of 7, resulting in a risk exposure of 2.45.

By integrating the risk assessment results, the planning team had derived additional objective information to help identify and prioritize the software issues. The risk-exposure calculations strongly supported the less formal definitions of the software issues, and helped to focus the measurement efforts of the program.

Captain McLain identified the following additional concerns:

- The budget was fixed for the program as a whole. Strict financial monitoring was a high priority. The program was required to report Earned Value for both hardware and software.

- With concurrent hardware development, the possibility of software requirements growth was almost a certainty as the software would have to make up for unforeseen hardware problems.

- A high level of supplier performance was also a key part of the strategy. Captain McLain felt that her primary leverage here was to choose a contractor with a mature life-cycle process and appropriate domain experience.

- Product quality was also a concern, considering that these were life-critical applications.

- At the completion of the PMO-551 planning process, Captain McLain had a good idea of what her software development issues were and where she would have to focus her attention during the development. Given the complexity of the DDG 51 SCWE program, she knew that measurement was her primary means for staying on top of things.

2.2 Software Acquisition

2.2.1 Request for Proposal

After the PMO-551 software cost and schedule estimate, the risk analysis, and issue identification were complete, Captain McLain turned her attention to issuing the DDG 51 SCWE Request for Proposal (RFP) and to choosing a capable supplier. The results of the design analysis studies were made available to all bidders. At the bidders' conference, Captain McLain made it clear that the successful bidder would have to demonstrate an effective life-cycle process. With the success of the program tied to the overall capability of the software supplier, Captain McLain specifically addressed her software development requirements in the RFP. In their proposals, the bidders were required to provide the following:

- Their preliminary allocation of the system level requirements (provided with the RFP) to a proposed software architecture

- Their approach for using the existing DDG 51 C4I software as the baseline for DDG 51 SCWE software development

- Their proposed use of COTS software components and an OSA in the redesigned system

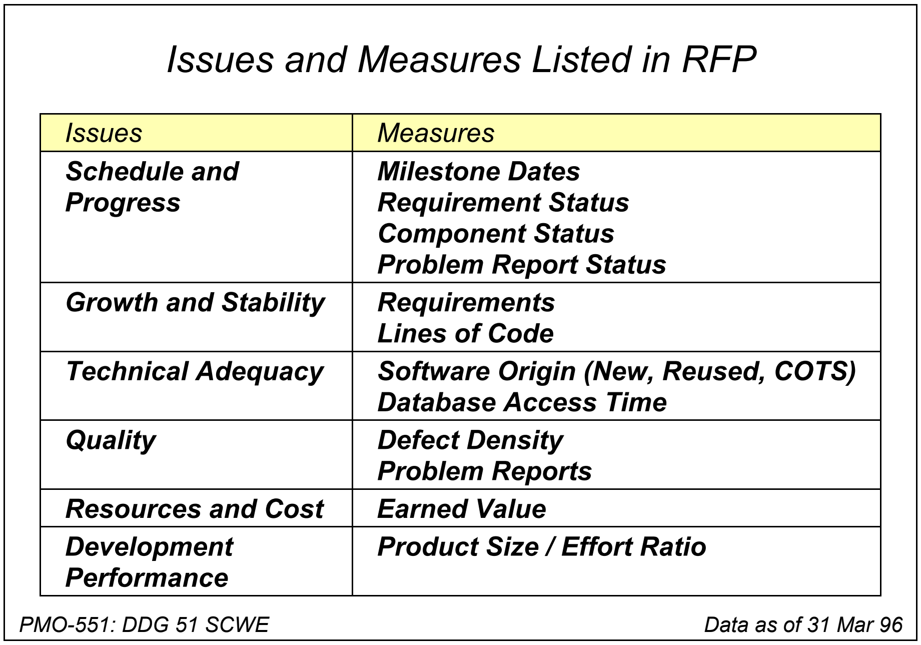
- A comprehensive set of software data describing the bidder's performance on similar development programs. These data included sizing, schedule, effort, and problem report data, as well as program descriptive data required to evaluate the supplier's software development performance.

- A proposed work breakdown structure (WBS), that would be used as a basis for tracking Earned Value

- A detailed description of the proposed software life-cycle processes and activities, coupled to an overall DDG 51 SCWE project management plan.

- A description of how measurement. risk management, and financial performance management would work together to objectively manage the project effort

The program issues were outlined in priority order as shown in Figure 8A-3. The RFP also listed the measures shown in the figure. The bidders were required to fill in detailed attributes for each measure based on what was available from their development process. For example, for the measure "component status," they were required to define "component" and to indicate what status would be tracked, such as design, code, and integration and test, along with the exit criteria.



Issues and Measures Listed in RFP

Issues	Measures
Schedule and Progress	***Milestone Dates*** ***Requirement Status*** ***Component Status*** ***Problem Report Status***
Growth and Stability	***Requirements*** ***Lines of Code***
Technical Adequacy	***Software Origin (New, Reused, COTS)*** ***Database Access Time***
Quality	***Defect Density*** ***Problem Reports***
Resources and Cost	***Earned Value***
Development Performance	***Product Size / Effort Ratio***

PMO-551: DDG 51 SCWE

Data as of 31 Mar 96

Figure 8A-3. Software Development Issues

2.2.2 Proposal Evaluation

The RFP was released in the fall of 1995, and a total of five proposals were submitted. Of these five, two were considered by the source selection team to be in the competitive range. Each of the two prime contractors on these bids was teamed with several subcontractors. After a detailed evaluation of each proposal, a recommendation for award was forwarded to the program manager. There were many aspects about the winning proposal that impressed the source selection team:

- The successful bidder’s historical data was credible. The proposal provided clear definitions for project size, schedule, effort, and problem report data, and indicated what was included and what was excluded in the numbers. The data supported the bidder’s claim that it had an effective life-cycle process.

- The successful bidder’s DDG 51 SCWE project management plan was based on achievable performance and productivity objectives, and the rationale for the projections was supported by objective estimates of the associated software parameters. Further, the proposed project management plan included a detailed WBS mapped to the proposed architecture and development activities. The WBS related the proposed development process to the bidder’s recommendations for tailoring MIL-STD-498.

- The successful bidder’s project management plan included an incremental development approach with a relatively sequential set of development activities allocated between two major increments.

- The successful bidder’s proposed measurement program met all of the requirements specified in the RFP and clearly reflected that the bidder had experience in using both measurement and risk management to support successful development programs.

From the systems design perspective, the successful bidder met the defined technical requirements for the DDG 51 SCWE program. The proposed system design, as shown in Figure 8A-4, included the following:

- The modification of the existing system architecture to include open system interfaces. This change called specifically for the implementation of an open, commercial standard Fiber Distributed Data

Interface (FDDI) Local Area Network (LAN) to interface the existing sensors and the new functions to the AN/UYK-43 computer. This design change provided for minimal "breakage" to the existing system and supported an affordable development and future system expansion using cost-effective components.

- The development and integration of new display workstations with integrated processors to handle the new, multiple-ship engagement functions and associated display and control functions. The proposed workstation design made use of both COTS hardware and software. The workstations were to be interfaced to the baseline system through the FDDI LAN. This approach also addressed the need for an advanced human-machine interface required to implement the new target engagement and weapons management functions.

- The replacement of the existing flat-file data management software in the AN/UYK-43 with a COTS-based relational database manager and the use of the same relational database structure for the new applications resident in the workstations. This design change addressed the large increase in the amount of data that the new system would have to process.

- The reallocation of the revised software functionality between the AN/UYK-43 and the new processors in the display workstations. The proposal included the revision and reallocation of the critical engagement and weapons management to functions in the workstation processor.

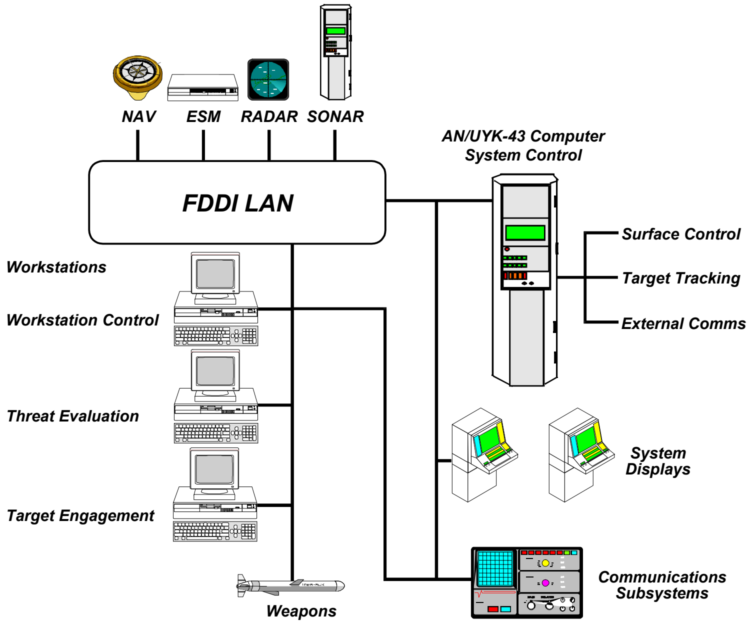


Figure 8A-4. Proposed Revision to Architecture

In all, the new software architecture added one major function, Workstation Control, to the system. The AN/UYK-43 Threat Evaluation function, however, was materially revised and moved to the workstation processor. The AN/UYK-43 Target Engagement function was to be completely rewritten and also moved to the workstation. This increased the number of CIs to 32. The overall amount of software change was significant, but it reflected the nature of the new concurrent weapons engagement mission requirements.

2.2.3 Award

The PMO-551 primary contact for project measurement, Gary Wilson, was a member of the source selection team. His analysis of the submitted project measurement data was an important factor in selecting the winning bidder. Another significant factor was the quality of the data in the winning proposal that demonstrated the supplier’s ability to objectively identify and manage software issues using project measurement.

The source selection team developed a number of project measurement indicators to support analysis of the proposed project plans. The critical question was the feasibility of the proposed software development schedule, given the bidder’s estimated project size and proposed effort profile. This assessment was based on the calculated software development productivity required to meet the proposed objectives and the relationship of this productivity to the bidder’s performance history. Did the proposal indicate, for example, that the bidder would have to improve his demonstrated software productivity significantly to meet his proposed schedule? If so, was his approach for doing this realistic?

Of equal importance was the relationship between the proposed DDG 51 SCWE software planning parameters. For example, did the scheduled software development activities peak while the development staff was being reduced? These were the types of questions the source selection team was asking.

Gary Wilson developed an indicator that showed the software productivity history of the two bidders in the competitive range. On the same indicator, he graphed the software productivity required for the DDG 51 SCWE, based on the measurement data submitted in each of the proposals (Figure 8A-5). The project size estimates were normalized based on how the supplier said the code was to be implemented (COTS, NDI, new, or modified), and the schedule and effort data was used as it was submitted.

The software productivity indicator clearly showed that the successful bidder had proposed a software productivity rate for the DDG 51 SCWE program that was consistent with his historical performance. The unsuccessful bidder had proposed a significant increase over his demonstrated productivity rate, but there was no basis for his claim. In fact, when the source selection team investigated, it found that the high productivity rate, as proposed, was tied to an artificially low-cost bid in terms of the number of staff planned for the development program. In addition, the historical data submitted by the unsuccessful bidder was inconsistent, with no clear definitions for how lines of code, effort, or milestones were measured. The source selection team requested several clarifications from the bidder, but did not receive enough information to substantiate the data.

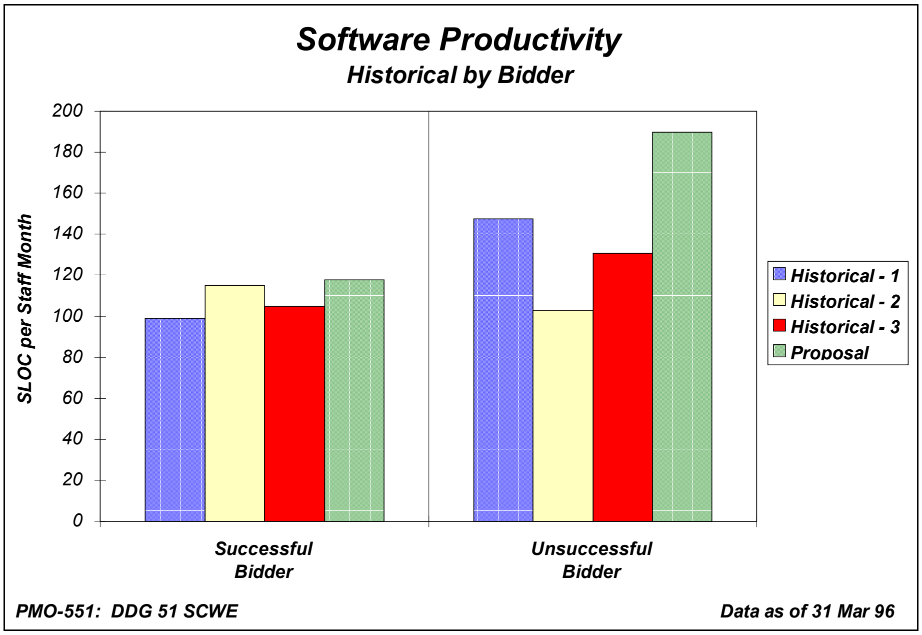


Figure 8A-5. Productivity Required for Each Proposal

One concern with the successful bidder’s proposal was a somewhat risky 60-month development schedule. The source selection team, however, felt that the life-cycle process, as proposed, was capable enough to mitigate this risk.

With respect to the cost and technical proposal evaluations, the project measurement results supported award to the higher priced, but more credible bidder. The successful bidder’s software data was clearly representative of an organization that had an established project measurement program embedded into a mature life-cycle process. This bidder’s measurement program could best address the software issues and risks associated with the DDG 51 SCWE program.

In May 1996, Captain McLain announced that CDX Systems, Inc., was awarded the development contract for the DDG 51 Surface Ship Concurrent Weapons Engagement Upgrade Program.

2.2.4 Negotiations

During contract negotiations, PMO-551 finalized the software development and measurement plans with the program manager from CDX Systems. There were several key objectives:

- Re-affirm the software development start date of July 1, 1996
- Define the software development schedule, effort, and sizing plans
- Define clearly which software measures would be applied, how CDX Systems would define each software measure, and how project measurement data would be transferred between CDX Systems and the program office
- Ensure that the subcontractors were consistent in their use of measurement when reporting to the prime contractor

The discussions with CDX Systems were extremely important. The supplier was able to make sure that the program office software team had a clear understanding of the software data they would be receiving. They would understand what the data represented, how it was measured, and, most important, how it related to the CDX Systems software development process.

Captain McLain asked her staff to evaluate the software plans for feasibility and consistency. Gary Wilson graphed a set of indicators based on the current CDX Systems planning data. These indicators are shown in Figure 8A-6, Figure 8A-7, and Figure 8A-8.

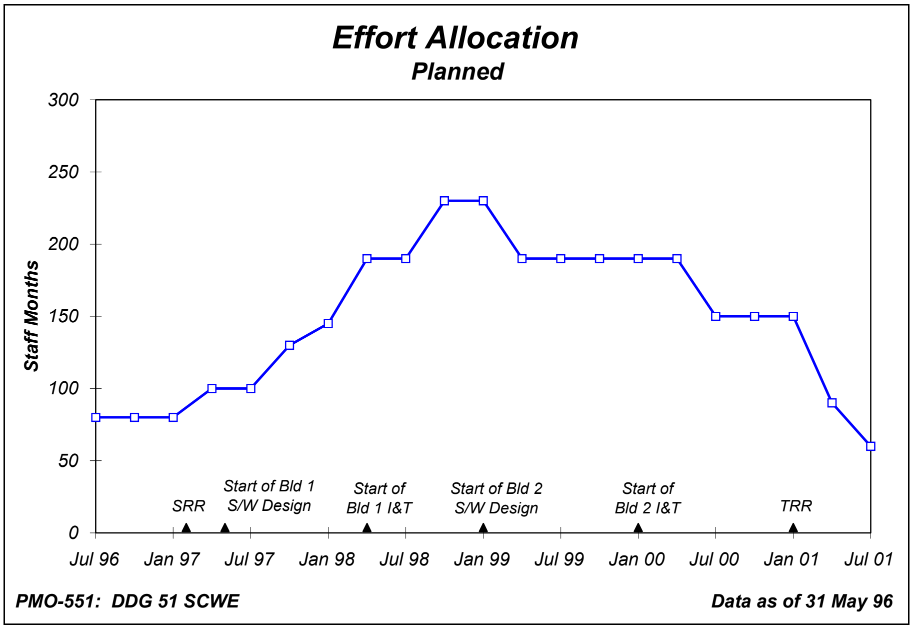


Figure 8A-6. Planned Effort Allocation

The proposed changes to the existing system resulted in a large increase in the total size of the software. Almost 700K lines of existing software were retained from the baseline system. Even with this amount of software reuse, close to one million new lines of code would have to be written. With the addition of the COTS components, the total estimated size of the new system was over 3 million logical lines of code. The project effort plan showed a traditional staffing profile and was consistent with the overall development activities as scheduled. Overall, software planning data represented a well-defined software development approach.

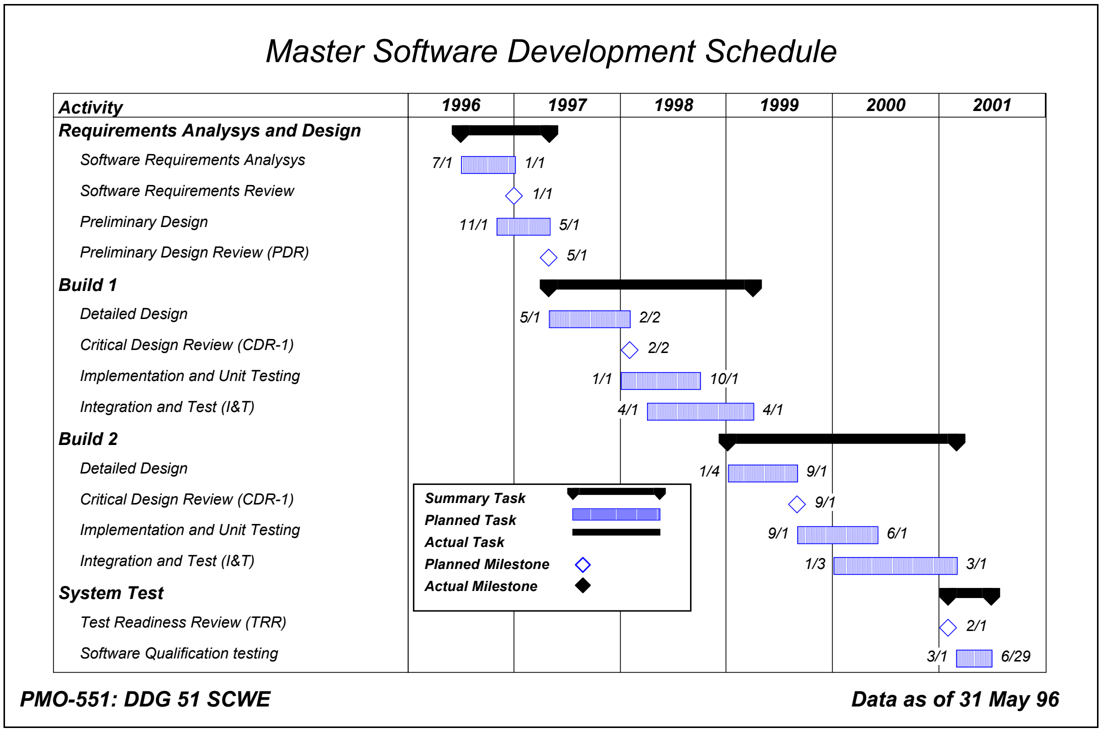
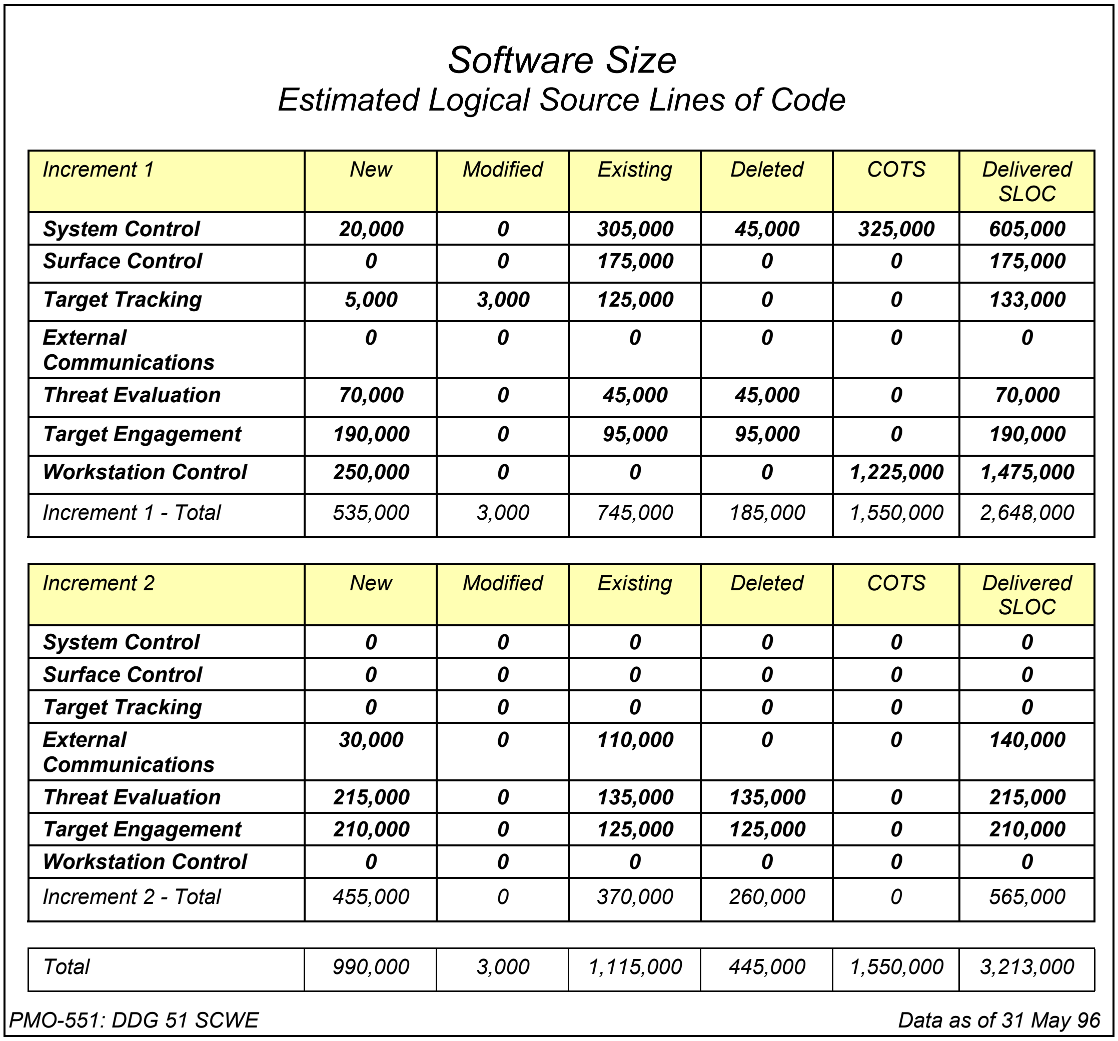


Figure 8A-7. Schedule



Software Size

Estimated Logical Source Lines of Code

Increment 1	New	Modified	Existing	Deleted	COTS	Delivered SLOC
System Control	**20,000**	**0**	**305,000**	**45,000**	**325,000**	**605,000**
Surface Control	**0**	**0**	**175,000**	**0**	**0**	**175,000**
Target Tracking	**5,000**	**3,000**	**125,000**	**0**	**0**	**133,000**
External Communications	**0**	**0**	**0**	**0**	**0**	**0**
Threat Evaluation	**70,000**	**0**	**45,000**	**45,000**	**0**	**70,000**
Target Engagement	**190,000**	**0**	**95,000**	**95,000**	**0**	**190,000**
Workstation Control	**250,000**	**0**	**0**	**0**	**1,225,000**	**1,475,000**
Increment 1 - Total	535,000	3,000	745,000	185,000	1,550,000	2,648,000

Increment 2	New	Modified	Existing	Deleted	COTS	Delivered SLOC
System Control	**0**	**0**	**0**	**0**	**0**	**0**
Surface Control	**0**	**0**	**0**	**0**	**0**	**0**
Target Tracking	**0**	**0**	**0**	**0**	**0**	**0**
External Communications	**30,000**	**0**	**110,000**	**0**	**0**	**140,000**
Threat Evaluation	**215,000**	**0**	**135,000**	**135,000**	**0**	**215,000**
Target Engagement	**210,000**	**0**	**125,000**	**125,000**	**0**	**210,000**
Workstation Control	**0**	**0**	**0**	**0**	**0**	**0**
Increment 2 - Total	455,000	0	370,000	260,000	0	565,000

Total	990,000	3,000	1,115,000	445,000	1,550,000	3,213,000

PMO-551: DDG 51 SCWE

Data as of 31 May 96

Figure 8A-8. Software Size Estimates

With the contract negotiations completed, Captain McLain felt that the program plan was feasible. She also felt that she had a clear picture of the program's software development issues:

- The real possibility for growth in the software requirements
- The adequacy and effectiveness of the software development technical approach
- The overall impact of cost and schedule constraints on the ability of the supplier to build quality into the software
- The adequacy of the supplier's life-cycle process capability

3

Software Development

After the DDG 51 SCWE contract was awarded, Captain McLain began the complex task of managing the software development process. Project measurement activities shifted from evaluating the software plans to tracking performance against those plans. With her own Navy program organization and CDX Systems, Captain McLain believed she had a capable development team - one that could effectively identify and resolve the expected development issues and make the program a success.

This chapter explains how project measurement helps identify and objectively analyze software issues and shows how the program manager uses the resulting information to make informed decisions. For the DDG 51 SCWE program, project measurement has become an integral part of the program management process and provides PMO-551 with an effective tool for communicating with the supplier.

3.1 Tracking Development Performance

3.1.1 Project Measurement Overview

DDG 51 SCWE software development officially started with the kickoff meeting between CDX Systems and PMO-551 on 1 July 1996. At the kickoff meeting, Captain McLain explained it was important that her engineering staff communicate effectively with the developers at CDX Systems. She also stated her expectation that they take an integrated team approach to resolving any technical and management issues. Captain McLain addressed the importance of an effective project measurement process and emphasized she would use the software data to help manage the program and to identify problems as early as possible.

CDX Systems presented an overview of their DDG 51 SCWE development process and explained how would use project measurement to manage the progress and quality of the software. The lead CDX software engineer on the program provided a description of the key characteristics of their measurement program:

- The overall measurement program was applied across all software development activities at the CI level. A single project measurement database would be created, with both the contractor and the program office having access. Both would be managing from the same data and both would have real-time, simultaneous access to changes. For some measures, such as lines of code, data was collected down to the level of individual units. Per the development contract, PMO-551 would have direct electronic access to this data.

- For the program, the process for estimating and measuring each software parameter was defined and was consistent with the CDX approach used for past programs. In addition, CDX reported that all of the software development subcontractors agreed to use the same measurement definitions.

- CDX Systems reviewed the DDG 51 SCWE software WBS and showed how the overall measurement structure was aligned with the defined software activities and products. They also reviewed their MIL-STD-498 implementation.

- CDX Systems stressed that the measurement process began with the accurate definition and tracking of both the stated and derived software requirements. They showed how they would measure the total number of requirements and how they would track the allocation of the requirements to the software architecture.

- CDX completed the discussion by reviewing the overall set of measures they intended to use. The measures themselves were relatively basic, but were implemented within a well-defined process at a meaningful level of detail.

3.1.2 Software Issue Identification and Analysis

During the first year, the program proceeded relatively smoothly. The project measurement process was in place and Captain McLain received a monthly issue evaluation from Gary Wilson. The project measurement indicators showed some variance in the monthly actuals relative to the plans, but there were no major deviations. The Preliminary Software Design Review, which addressed the CI architectural design, was completed on 15 June 1997, six weeks behind schedule. Considering the DDG 51 SCWE was a six-year development program, this was only a small schedule slip and did not cause much concern.

In June of 1997, the Navy decided that the DDG 51 SCWE functional baseline had to be modified to incorporate a new variant of the surface-launched Tomahawk cruise missile. The functions required to implement this new missile were added to the Increment 1 software requirements. Since the new missile was added at the beginning of CI detailed design, both PMO-551 and CDX Systems believed that there would not be any major schedule impact from the modification.

During the next two months, the engineering IPT analyzed and documented the additional requirements and prepared the technical inputs for the Engineering Change Proposal (ECP). The new Tomahawk variant added approximately 550 additional requirements and 62,000 source lines of code (SLOC) to the planning baselines. The resultant changes were allocated to 450 new software units. The majority of these requirements were applicable to the Target Engagement function. The Workstation Control and System Control functions also had minor revisions due to the new missile. These new requirements increased both the schedule and technical risk associated with the Target Engagement function, which had already been assessed as high by both PMO-551 and CDX Systems.

In November of 1997, Gary showed Captain McLain an Increment 1 development progress indicator based on the number of software units completing detailed design (Figure 8A-9).

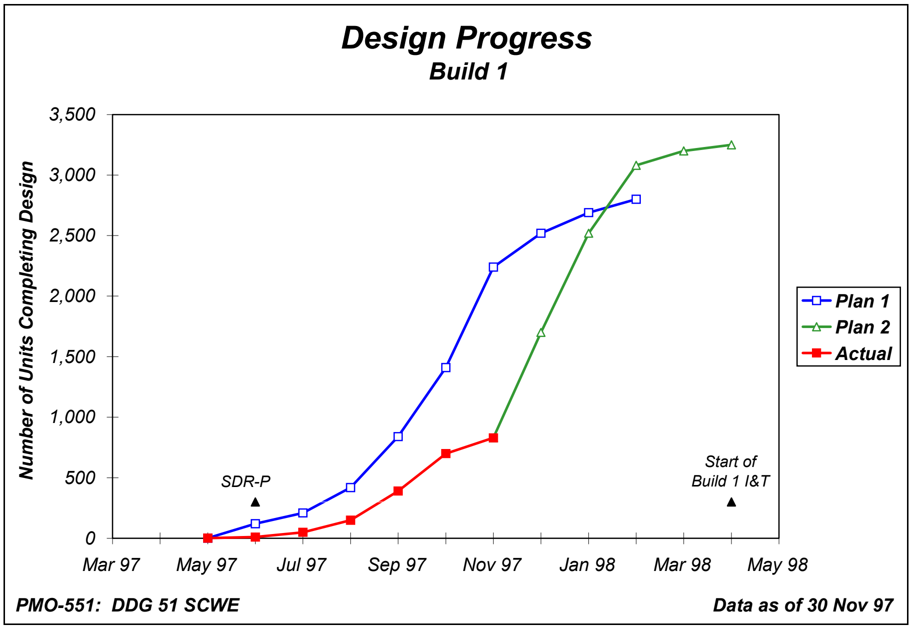


Figure 8A-9. Software Development Progress

The first thing he pointed out was the lag in development progress. The number of units completing detailed design was significantly behind plan. Yet the Earned Value reports (Figure 8A-10) and the milestone (Gantt) charts (Figure 8A-11) showed that the program was close to being on schedule. There were minor negative cost and schedule variances in the Earned Value data, but these were not large enough to cause any concern. It later became clear that the Earned Value was based on too high a level of WBS to provide early indications of schedule problems. The WBS that CDX Systems had been tracking against for Increment 1 is shown in Figure 8A-12.

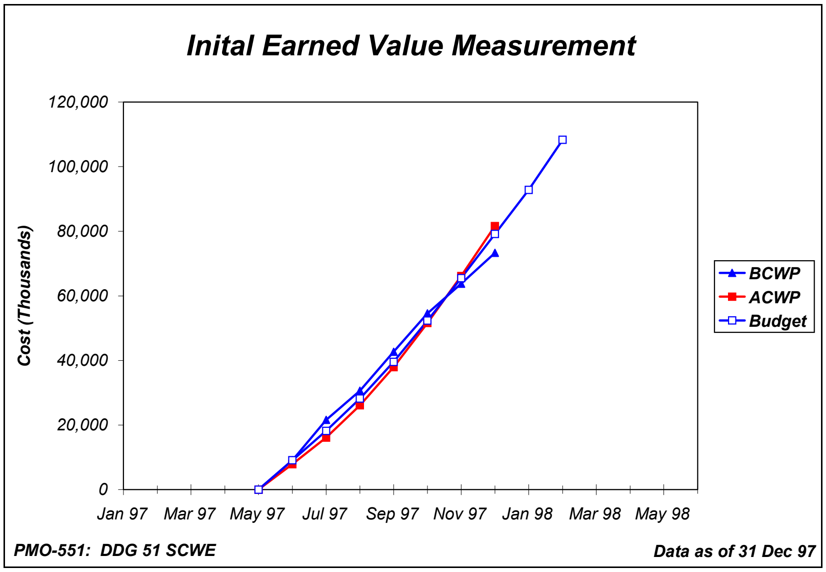


Figure 8A-10. Earned Value

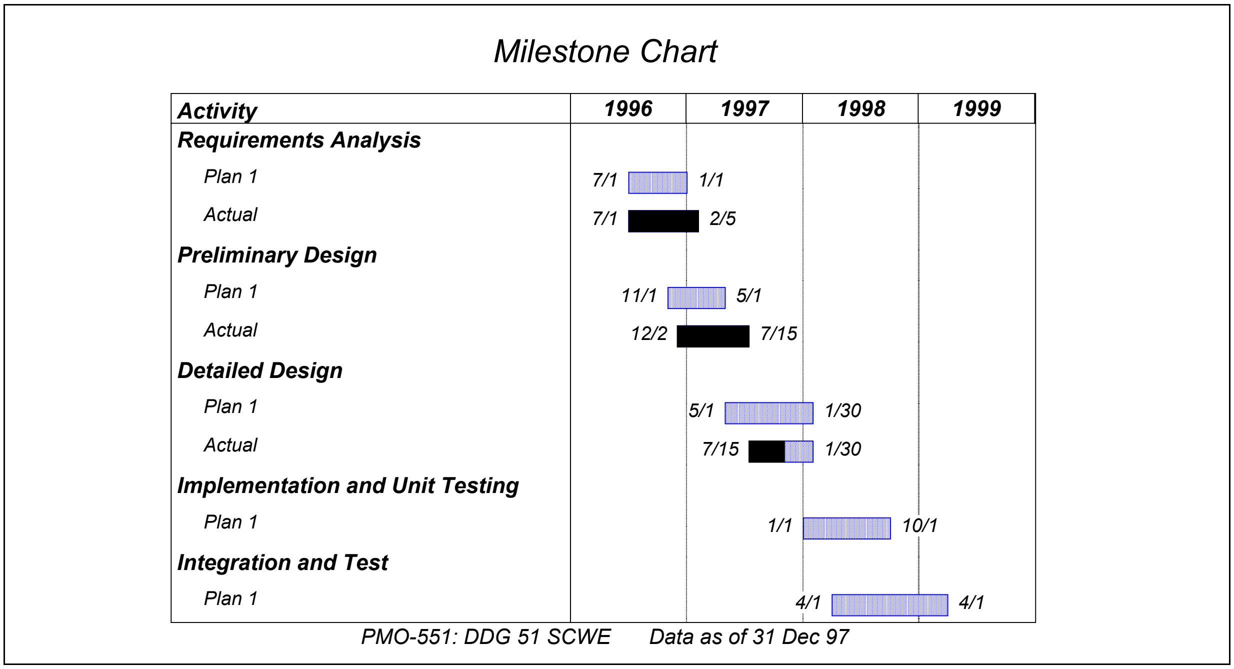


Figure 8A-11. Schedule Milestone

1.0 DDG System
1.3 DDG Software
1.3.1 Increment 1
1.3.1.1 System Control
1.3.1.1.1 Requirements Analysis
1.3.1.1.2 Design
1.3.1.1.3 Coding and Module Testing
1.3.1.1.4 Component Integration and Testing
1.3.1.1.5 CI Testing
1.3.1.2 Surface Control
1.3.1.3 Target Tracking
1.3.1.4 External Communications
1.3.1.5 Threat Evaluation
1.3.1.6 Target Engagement
1.3.1.7 Workstation Control
1.3.1.8 CI to CI Integration and Testing
1.4 System Engineering
1.4.1 Requirements Engineering
1.4.2 System Architecture
1.4.3 Configuration Management
1.4.4 Release Engineering
1.4.5 Rework
1.4.6 Security Certification and Accreditation
1.5 Testbed
1.5.1 Equipment
1.5.2 Software
1.5.3 Operation
1.6 System Qualification Test
1.6.1 Test Plans
1.6.2 Test Equipment and Software
1.6.3 Test Operations
1.7 System Test and Evaluation
1.7.1 Development Test and Evaluation
1.7.2 Operational Test and Evaluation
1.8 Operational Site Activation
1.8.1 Installation/Integration of S/W System

Figure 8A-12. WBS

There was also a comparable WBS for Increment 2. Each of the seven configuration items was broken down into the activities of requirements analysis, design, coding and module testing, component integration and testing, and CI testing (In the interest of clarity, that breakout is shown only for the System Control CI). CDX Systems received 50 percent credit when an activity began and the remaining 50 percent credit when it finished. Using this approach, CDX could start many tasks and appear to be on or ahead of schedule. Since design had begun for all CI's, progress appeared to be on track. The more detailed information provided by the work unit progress measure showed that the detailed design of Increment 1 was behind schedule.

Captain McLain tasked the engineering IPT to work with Gary on a more detailed work breakdown structure that could be used to track Earned Value in a way that would more accurately reflect schedule and cost deviations at any point in time. They allocated budgets down to the level of individual software units. The engineering IPT worked with each of the configuration item managers to allocate budgets to that level of detail. An additional three-digit code was added to the time reporting system to identify each unit. With the more detailed information, the Earned Value Measurement System was revamped to provide less credit for entering the design phase and more credit given to the other areas of the design activities. As a result, a new report of Earned Value was developed as shown in Figure 8A-13. This new information identified that the detailed design was over three months behind schedule and about 40 percent over cost.

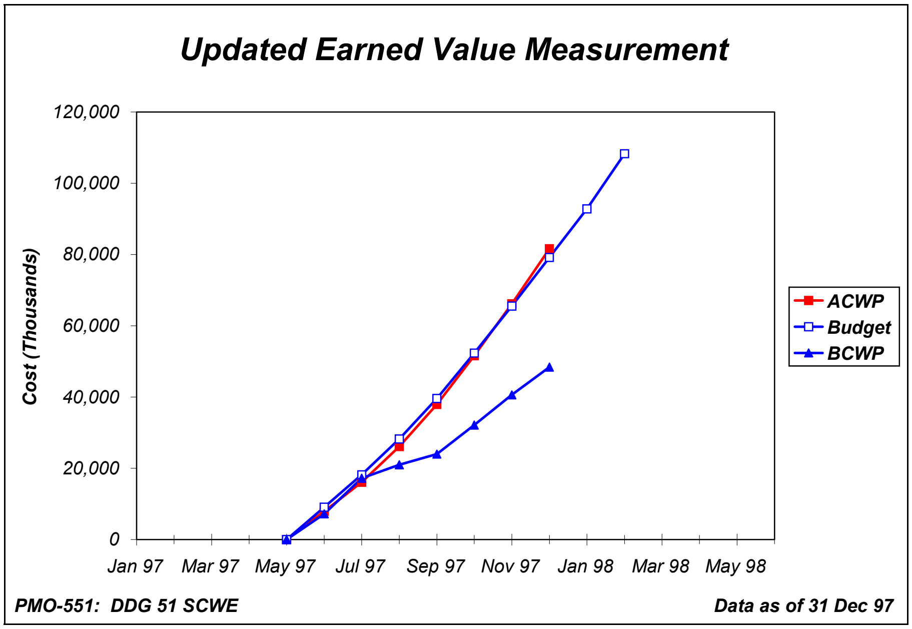


Figure 8A-13. Revised Earned Value

CDX Systems had developed a revised plan that took into account the additional software units that were added because of the new missile functionality. The new plan called for a much higher unit design completion rate than was originally projected or had been achieved to date.

Gary had discussed this indicator with CDX Systems. They believed that they could meet the higher unit completion rate projected in their revised plan. They based this assumption on the fact that they had added 80 new people to the staff over the past few months. CDX Systems indicated they now had sufficient resources available to complete the software development within the projected schedule.

In April of 1998, the supplier experienced serious and unexpected problems trying to integrate the COTS products into the DDG 51 SCWE software baseline. Specifically, CDX Systems ran into the following difficulties:

- The task of integrating the COTS operating system was considerably more complicated than had been originally anticipated. Performance problems required the design and implementation of a functional software "shell" between the applications software and the COTS operating system. This meant that new requirements and code had to be added to the Workstation Control function.

- Performance problems were discovered while integrating the COTS relational databases in both the workstation and the AN/UYK-43. The critical ship-to-ship data items were not being processed quickly enough. The only solution was to revert back to flat file processing for the critical portions of this data.

With this new set of problems, it was clear that the schedule problem was getting worse. In fact, the Increment 1 Software Design Review covering CI detailed design was delayed for almost three months.

Captain McLain continued to review the summary level indicators on a monthly basis. In August 1998, she decided she wanted to see some indicators that could localize the problem areas to specific software functions. She directed Gary to take a close look at the current set of indicators to assess program status.

First, Gary constructed a graph, shown in Figure 8A-14, showing the growth in requirements over the past two years. The first point, July 1996, represents the number of stated requirements that were defined in the contract proposal. Between the beginning of the contract and June 1997, the number of requirements increased. The majority of this growth occurred during software requirements analysis, as the CDX system and software engineers achieved a better understanding of the system functionality and developed the derived software requirements. The number of requirements increased again between June and August 1997 due to the addition of the new Tomahawk missile functionality. Between August 1997 and August 1998, the number of requirements again increased with the addition of requirements resulting from the problems experienced while integrating the COTS software.

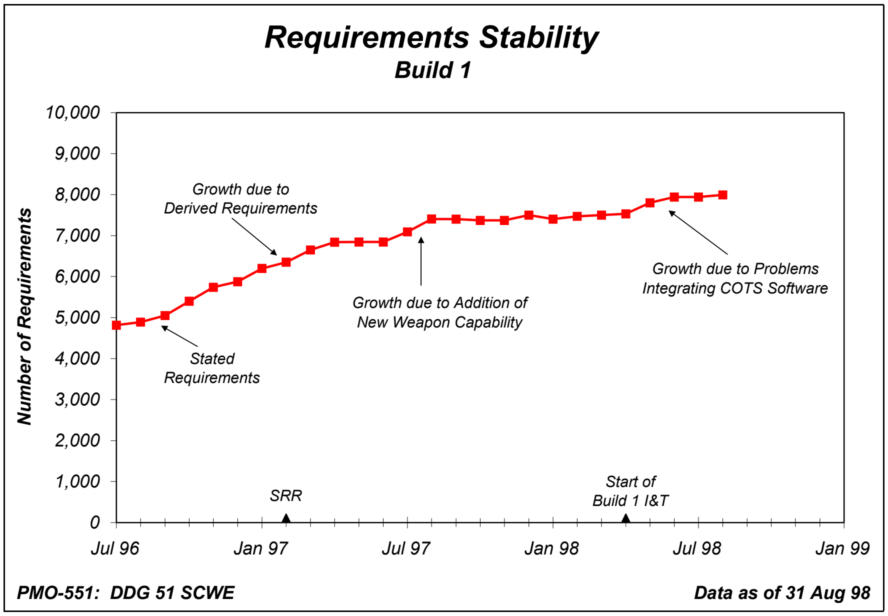


Figure 8A-14. Software Requirements

While it was obvious that the system as a whole experienced significant requirements growth, Gary also looked at the requirements growth for each of the major software functions in the system, as shown in Figure 8A-15. From this breakdown, it became clear that a large portion of the requirements growth was in the workstation functions. Most of the requirements growth related to the new missile occurred in the Target Engagement function. The growth related to problems with the COTS implementation had increased the number of requirements in the Workstation Control and System Control functions.

Captain McLain also wanted more information about the growth of requirements and the impact of that growth on product size. From the source data in the PMO-551 measurement database, Gary constructed a project size estimate by software origin indicator as shown in Figure 8A-16. This indicator showed that the requirements changes had not only increased the projected size of the software, but also had impacted how much of the software had to be newly developed. The latest plan indicated that more effort and schedule would be required due to an overall decrease in the estimated amount of non-developed code.

Captain McLain was also concerned about whether CDX's staffing levels were tracking to plan and if the amount of effort being applied to the program was adequate. The next graph Gary showed Captain McLain was the monthly effort data presented in Figure 8A-17.

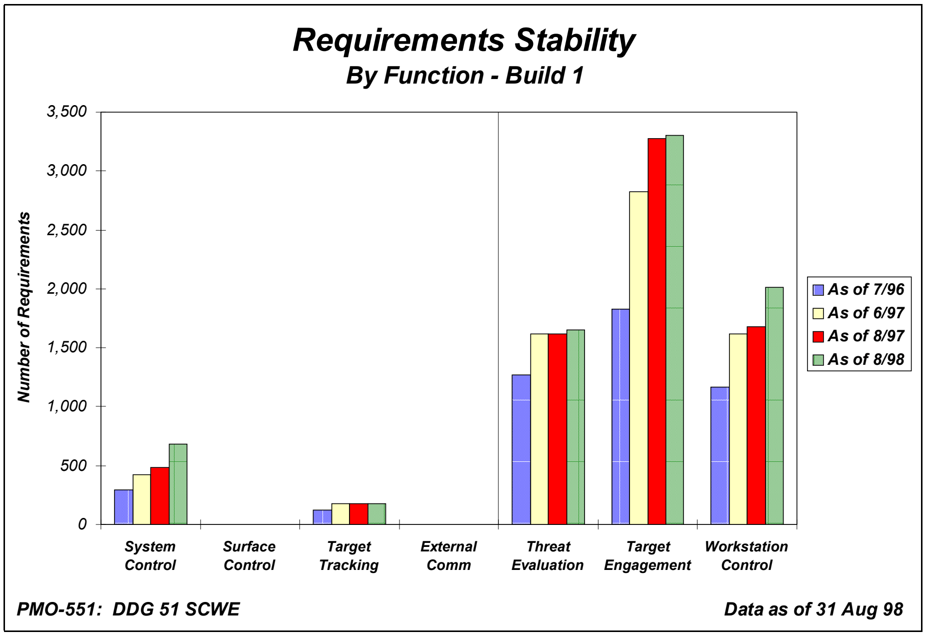


Figure 8A-15. Requirements Growth Versus Software Functions

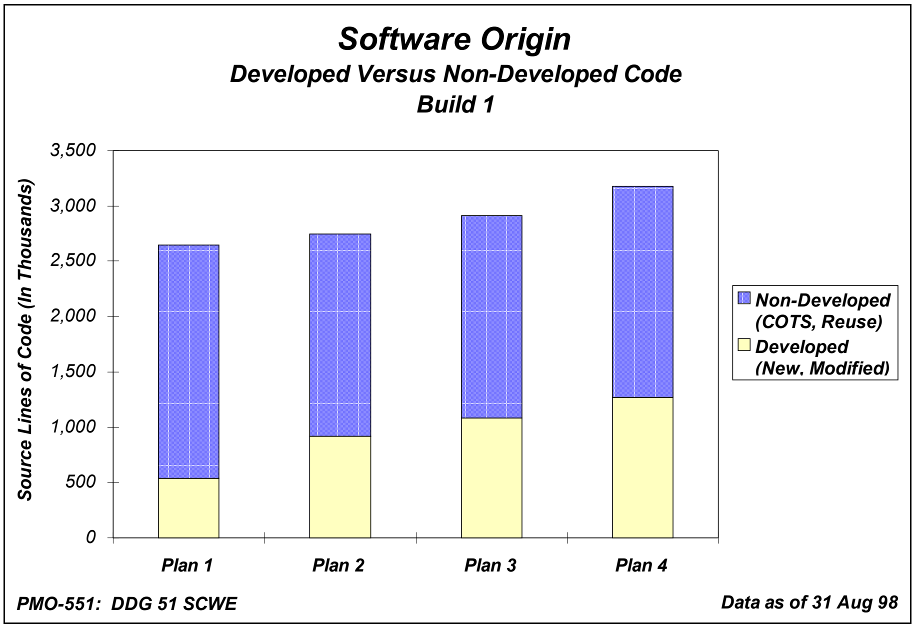


Figure 8A-16. Size Estimate By Software Origin

This graph showed that although the development was initially understaffed, CDX Systems added additional people to make up for the early deficit. In a subsequent discussion with CDX Systems, Captain McLain was assured there were enough people to complete the software development.

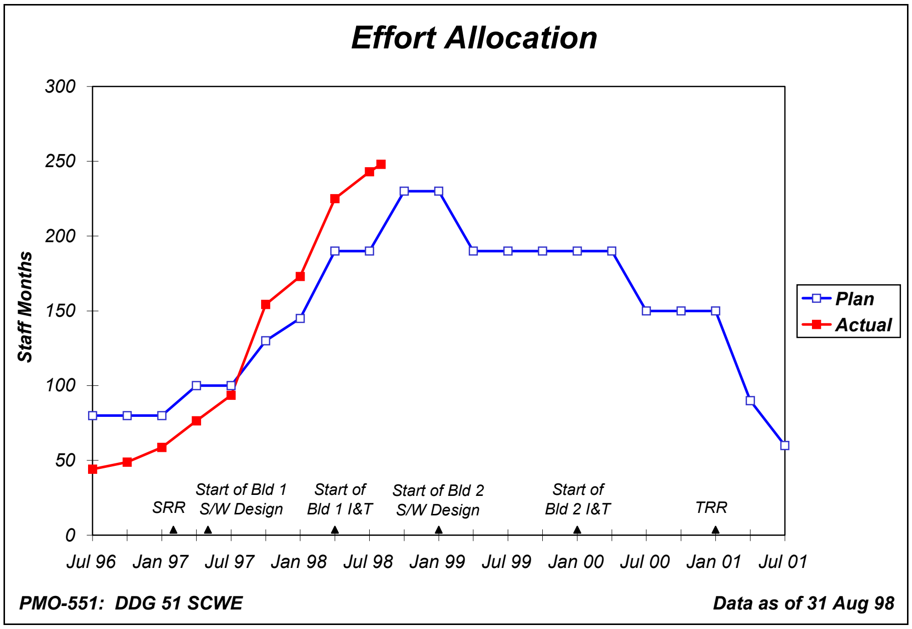


Figure 8A-17. Effort Allocation

Gary then showed Captain McLain an earlier indicator of software development progress based on the number of units that had completed detailed design, as shown in Figure 8A-18. From this indicator, it appeared the rate of units completing detailed design had increased significantly after the initial lag noticed in November of 1997. The data showed that all of the units had completed the detailed design milestone within one month of the revised plan.

While the progress indicator gave Captain McLain some reason for optimism, the problem report data told a different story. Gary showed Captain McLain a summary of the cumulative number of total and closed problem reports that had been collected during integration and test. These are shown in Figure 8A-19.

Captain McLain noted that the problem report discovery rate increased rapidly during integration and test. She was disturbed by the fact that problem report discovery appeared to be occurring at a much higher rate than problem report closure. She then asked Gary to show her the problem report data for the individual functions.

Gary calculated defect density by dividing the number of unique valid defects by the new and modified source lines of code for each function, as shown in Figure 8A-20. It was clear that, even when normalized by size, the Target Engagement function was much more of a problem than any of the other functions. Captain McLain asked Gary to find out what was going on with this function.

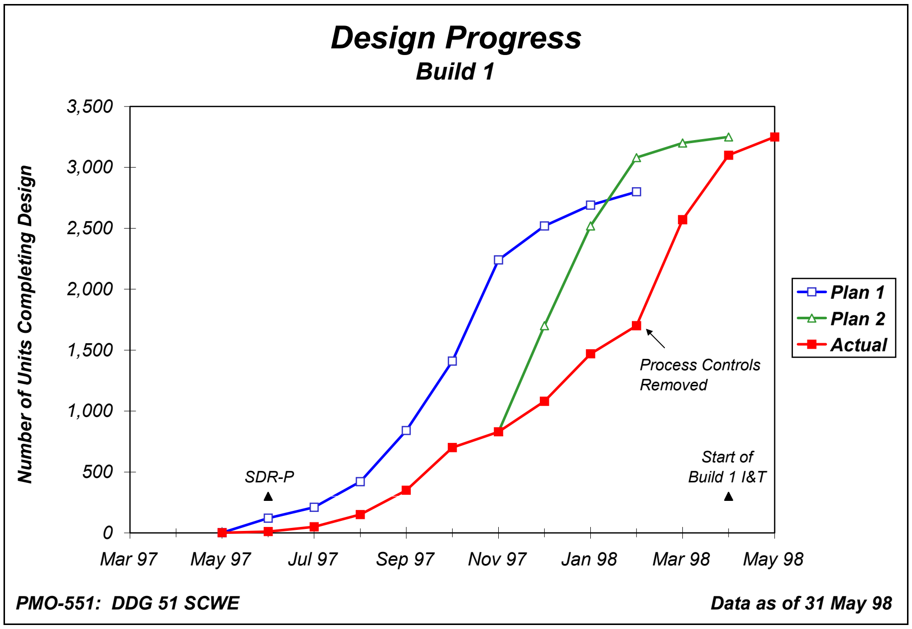


Figure 8A-18. Design Progress

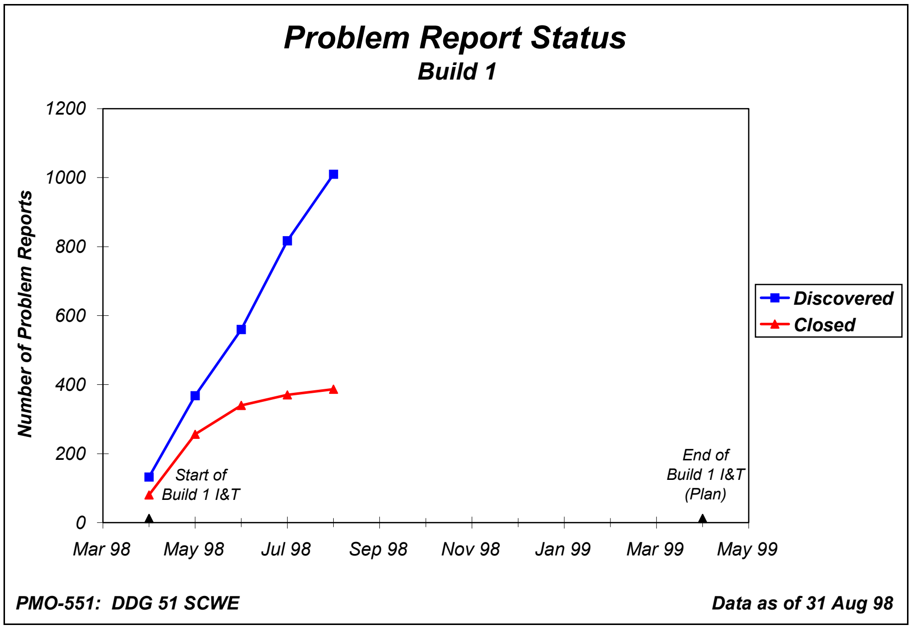


Figure 8A-19. Problem Report Status

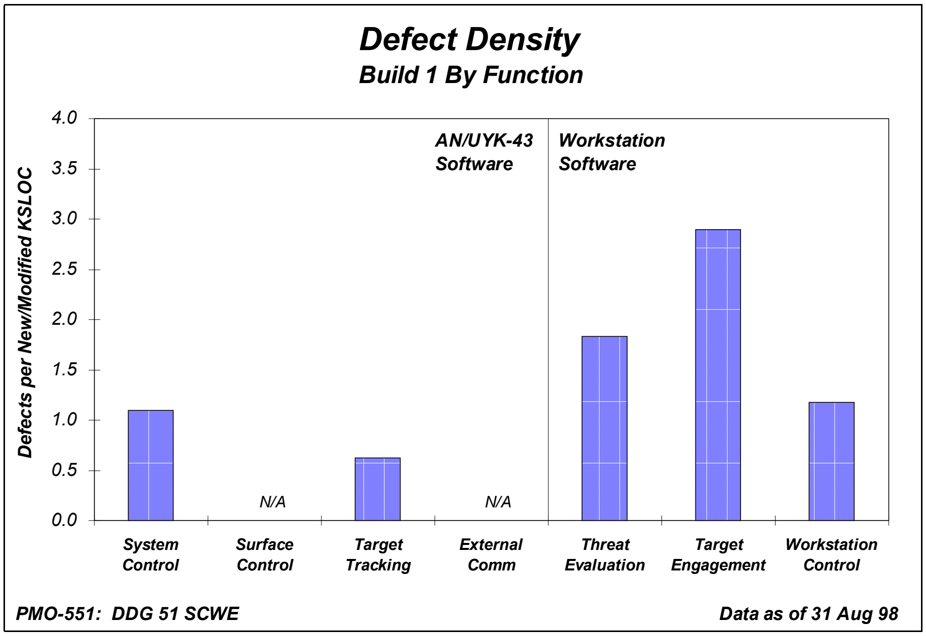


Figure 8A-20. Defect Density

Gary visited CDX Systems and met with the engineering manager to discuss the workstation CI problems. He discovered that, as of January 1998, unit design and code inspections had been discontinued in an effort to complete the development activities as quickly as possible. The delays in development progress had begun to impact the testing process. Successful completion of the unit design and code inspections had been the primary exit criteria for measuring unit development progress. With this requirement relaxed, the software suppliers were not required to adhere to a key process activity that would ensure that only complete, high-quality units were delivered for integration. *In effect, the quality of the software became secondary to meeting the schedule and the measurement indicators had helped to identify the problem.*

Most of the software units impacted by this process change belonged to the Target Engagement function. This explained the sudden increase in apparent development progress based on the number of units completing detailed design. It also explained the large number of problem reports being discovered in integration and test. Defects that should have been found during those inspections were not being discovered until later. In his discussions with CDX Systems personnel, Gary also found out that the majority of the recently added personnel were assigned to problem corrections and rework. It was clear that, by removing the process controls, CDX had only made the situation worse.

By this time, Captain McLain had serious doubts about the likelihood of completing Increment 1 on schedule. In examining the schedule revisions, as shown in Figure 8A-21, she noted that CDX Systems made a series of periodic minor revisions to the DDG 51 SCWE detailed milestone schedule. No changes to any intermediate milestone were made until it was obvious that the completion date for that milestone would not be met. Even when revisions were required, CDX Systems made only small incremental changes, rather than doing a comprehensive analysis to determine when the activities could realistically be completed. While completion estimates for the detailed milestones had slipped, the completion of integration and testing for Increment 1 had not been adjusted. The result was an integration and test schedule that was becoming less and less realistic.

Captain McLain realized that she might have to reassess the current DDG 51 SCWE project management plan. Her suspicions were confirmed when she looked at the achieved productivity to date for Increment 1. It did not appear that CDX Systems would be able to produce the planned amount of code for Increment 1 within the current schedule. After reviewing the analysis results with the CDX Systems program manager, Captain McLain decided to replan the software development effort.

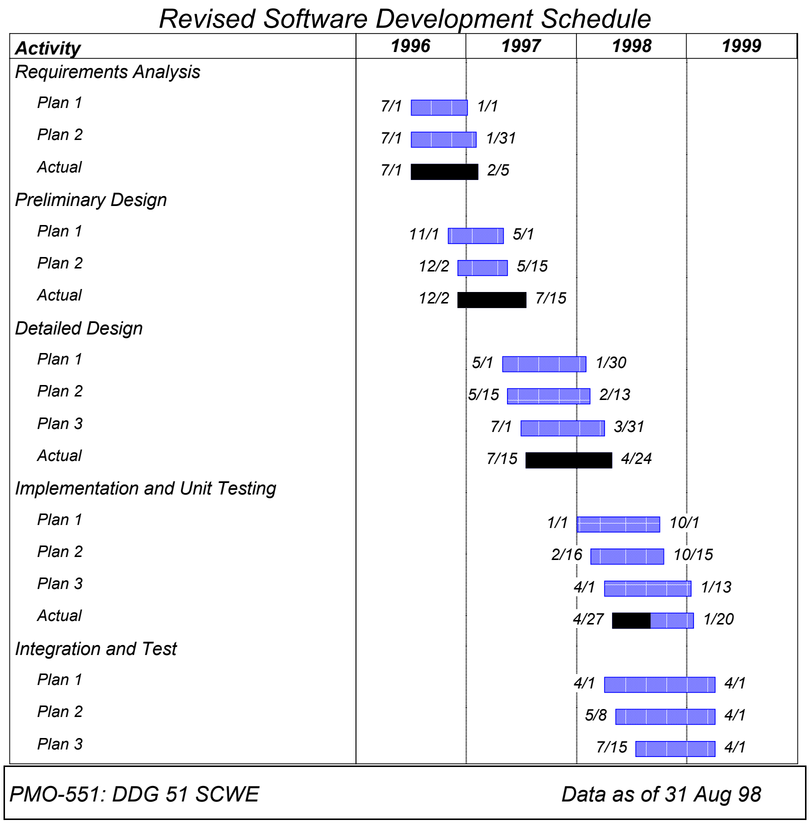


Figure 8A-21. Integration and Test Schedule

3.2 Revising The Development Plan

In October of 1998, Captain McLain met with the PMO-551 staff and with managers from CDX Systems to replan the remainder of the DDG 51 SCWE project. Two options were considered:

- Moving software functionality from Increment 1 to Increment 2.

- Adding another increment, Increment 3, and balancing software functionality between the increments. Under this option, Increment 1 and Increment 2 were revised so that each would contain an equal amount of code, while a smaller amount of code was integrated into Increment 3. Most of the code that was shifted to a later increment was from the Target Engagement function. This was the highest risk function and had the most problems with respect to development progress and quality. The Threat Evaluation and Workstation Control functions also had a small amount of code shifted between the increments.

Captain McLain asked Gary to evaluate the feasibility of each of these two options as shown in Figure 8A-22.

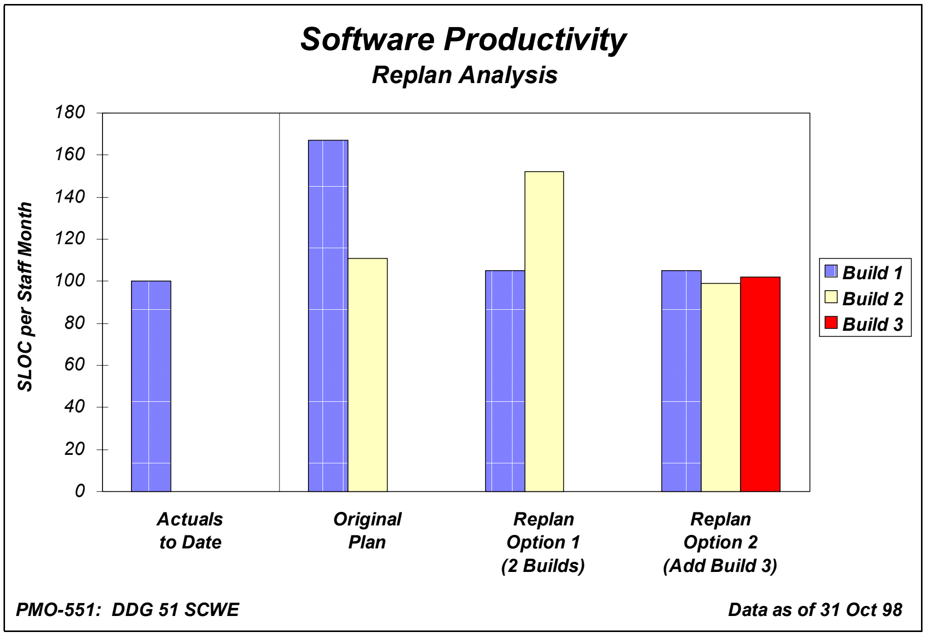


Figure 8A-22. Software Productivity

The first replan was rejected because of the high productivity requirement for Increment 2. This option required the productivity of Increment 2 to be higher than what had been achieved to date on Increment 1. This requirement was assessed to be unrealistic. Implementing this option would most likely have resulted in a second replan later in the development cycle.

The second option was selected because of several favorable elements:

- The required productivity for each remaining increment was based on CDX Systems' achieved software productivity to date on DDG 51 SCWE.

- This option supported the original delivery schedule of 1 July 2001, although with reduced functionality. An additional delivery was added for September 2002, 14 months after the original delivery. This additional delivery would include all of the required functionality.

- This option was based on the current staffing resources available to the DDG 51 SCWE program. No additional personnel would be required for this approach. It was believed that adding more people at this point in the development would only delay the delivery further.

- Although the schedule was extended by 14 months and additional funding had to be identified, the revised plan was realistic and contained no major risks.

When PMO-551 presented the replan to the Navy, the measurement analysis results helped to clarify the situation and showed that PMO-551 had an objective understanding of the software development constraints and issues.

As part of the replan, CDX Systems assured the program office that all process controls would be reinstated, including the unit design and code inspections.

3.3 Software Delivery

After the replan, Captain McLain and Gary continued to monitor the DDG 51 SCWE project. Captain McLain believed that two issues needed to be monitored more closely. First, she wanted to ensure that the requirements were being verified at a sufficient rate to meet the delivery schedule. Secondly, Captain McLain wanted to assess the adequacy of CDX Systems' integration and testing process.

To address the requirements issue, an indicator depicting the number of software requirements that had been successfully verified during integration and test was developed as shown in Figure 8A-23. Progress was steady, which told the program office that the planning revisions were effective. CDX Systems was producing the software in accordance with the revised schedule and was projected to meet all delivery requirements.

The quality of the software had also improved. The problem report discovery rate had begun to decrease, and even with the increased test activity, CDX was finding fewer serious problems as shown in Figure 8A-24.

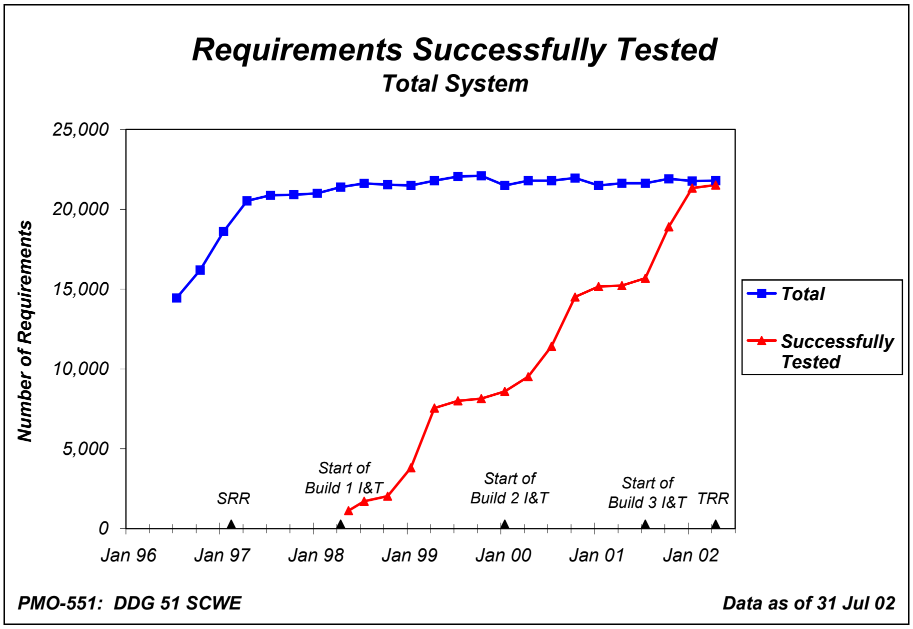


Figure 8A-23. The indicator for the number of software requirements successfully tested showed that the planning revisions were effective

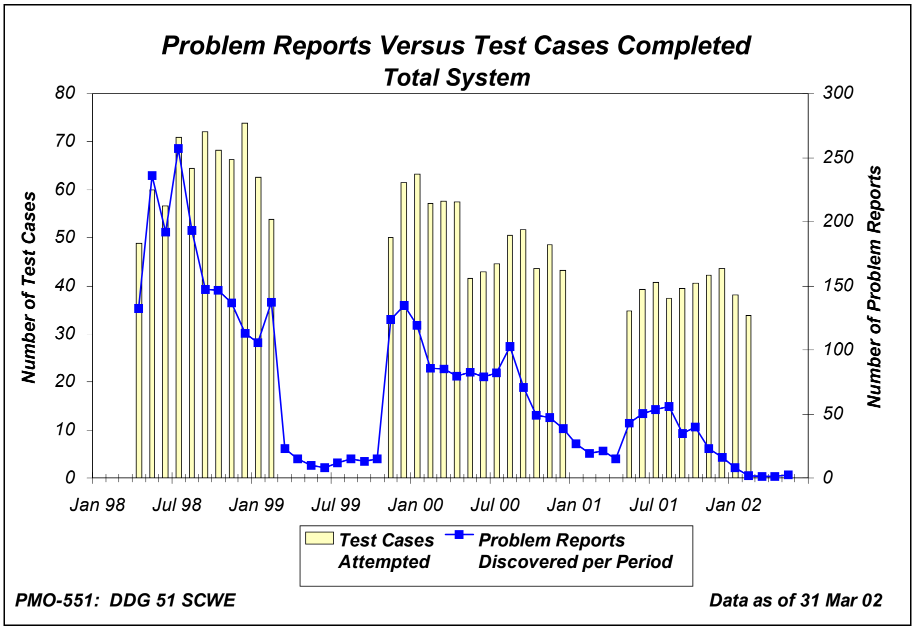


Figure 8A-24. The problem report discovery rate verified that the quality of the software had also improved

3.4 Epilogue

Increments 1 and 2 were delivered on schedule. Figure 8A-25 shows the actual productivity achieved for those increments, along with the productivity observed for Increment 3 as of March 2002. Actual productivity increased across the three increments, contributing to the on-time delivery for each increment.

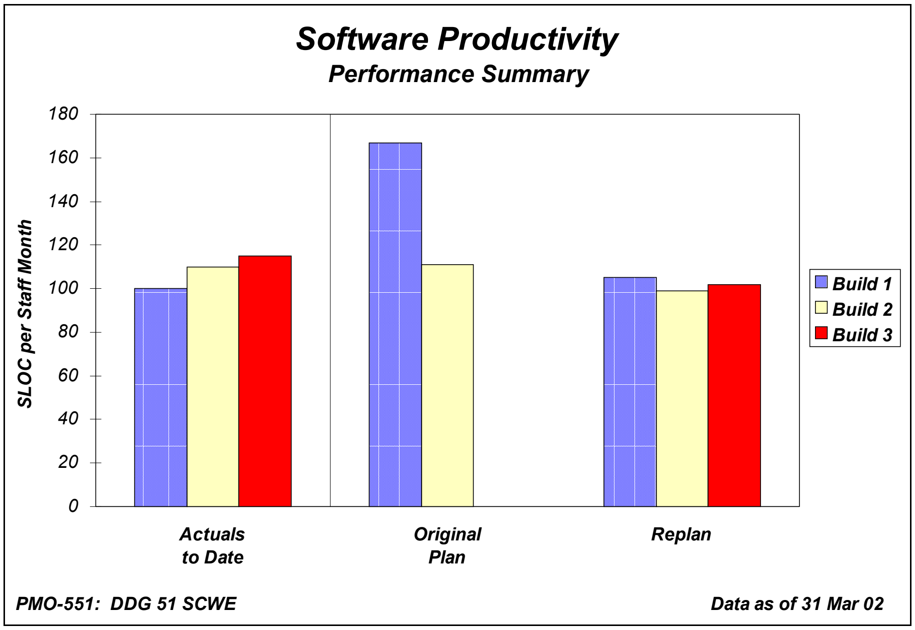
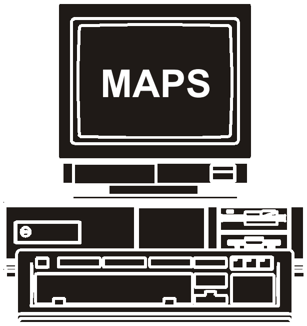


Figure 8A-25. The productivity achieved in each successive software increment improved

In September of 2002, the PMO-551 / CDX Systems, Inc. team deployed the **DDG 51 Surface Ship Concurrent Weapons Engagement Upgrade** system on the USS John Paul Jones, DDG 53. Although the issues related to the software development were significant, use of project measurement had helped the program manager to make objective and informed decisions that led to the program's ultimate success.



PART 8B

Information System Case Study

The Information System Case Study describes the development of a military personnel information system for the U.S. Air Force. This example demonstrates the use of measurement on a project that has been under development for some time. The project recently failed a major acquisition milestone review, and measurement is seen as a way to gain an increased level of control over the software development effort. The system is being developed by an organic Air Force activity working for a Program Manager within the same command. The development addresses current DoD initiatives to promote open systems, interoperability, and the use of commercial-off-the-shelf (COTS) software packages. The technical approach includes the use of multiple languages, including code generators, and conversion of existing data structures. The critical issues are largely driven by external development dependencies. They include the need to meet aggressive development and deployment schedules, and the requirement that the overall readiness of the software for deployment be objectively determined.

The Information System Case Study is organized into four chapters:

- **Chapter 1, Project Overview,** describes the technical and management aspects of the software development effort.

- **Chapter 2, Getting the Project Under Control**, shows how measurement can be implemented on an existing project to define a realistic project management plan, and how to track the development against that plan.

- **Chapter 3, Evaluating Readiness for Test,** illustrates how measurement helps to objectively determine if the software is ready for operational test and subsequent deployment.

- **Chapter 4, Installation and Operations and Maintenance,** shows how measurement is used after the system is fielded to identify and correct user problems.

1

Project Overview

This chapter introduces the project scenario and illustrates the technical and management aspects of the development effort. The project scenario describes the implementation of a measurement process on an existing project. Special consideration is given to using measurement data that is readily available within the established software and project management processes. The project is representative of a typical information system under development to meet business process reengineering objectives.

1.1 Introduction

Over the past several years, Ridgway Air Force Base in Cheyenne, Wyoming, has become established as a primary source for the development of Air Force business information systems. The software development group at Ridgway began as an organic software maintenance organization, and it has successfully transitioned its business base from the support of Air Force logistics and administrative systems to system reengineering and development. Ridgway has benefited from the recent DoD emphasis on upgrading existing information systems into an integrated set of more manageable, cost-effective resources, and has become an important resource in the Air Force Materiel Command.

In 1996, the Air Force designated Ridgway Air Force Base as the lead development organization for the Military Automated Personnel System (MAPS). MAPS represented the Air Force's “next generation” military personnel information system. The project was part of a larger initiative to reengineer the Air Force's administrative business processes. The reengineering plan included service-wide initiatives to modernize information system hardware, software, and communications interfaces at both the base and headquarters levels. Existing mainframes and terminals were to be replaced by client/server architectures, and new capabilities were to be implemented by adapting existing databases and integrating them with newly developed applications software. MAPS was an important link in this business system modernization effort, since it was the first part of the overall system to be developed and delivered. MAPS was scheduled to be deployed at a number of Air Force bases during 2000. Needless to say, MAPS was an important, and highly visible project.

In 1998, MAPS had been under development for two years. During that time, the Ridgway software development group had tried to keep current with changing DoD acquisition policy and related software initiatives. These included the definition of open systems architectures, the integration of commercial-off-the-shelf (COTS) software components, and the use of advanced programming languages and tools.

In November 1998, a new Program Manager was assigned to the MAPS project. Air Force Lt. Col. Barry Thompson was a 1981 graduate of the Air Force Academy. His background included four years with the Air Force's Operational Test & Evaluation Center and eight years in various Air Force system program offices. His last assignment was as the Deputy Program Manager for a major upgrade to an Air Force system that stored and processed maintenance records for aircraft.

Lt. Col. Thompson’s assignment to the MAPS project did not come under the best of circumstances. At the time of his arrival, MAPS had just undergone an unsuccessful review by the DoD's oversight committee for major information systems, the Major Automated Information Systems Review Council (MAISRC). MAPS had failed to receive a Milestone III approval for system production and deployment. This was largely a result of problems with the software, especially with respect to the amount of completed functionality and the overall quality of the existing code. The MAISRC report indicated that there was little confidence in the cost and schedule estimates presented by the previous Program Manager in an effort to substantiate his development plan. There was also a lack of available data to show the MAISRC how the Program Manager was addressing the key MAPS software development issues.

Lt. Col. Thompson arrived at Ridgway with clear direction to get the project under control and to establish an objective, credible plan for the remainder of the development. Lt. Col. Thompson's first task was to review the overall technical and management characteristics of the project. He wanted to identify the events and decisions that had helped to shape the project in order to identify the key software issues and problems that he needed to address.

1.2 Air Force Business Process Modernization Initiative

In reviewing the MAPS project history with the Ridgway development team, Lt. Col. Thompson learned exactly how MAPS fit into the Air Force Business Process Modernization Initiative. MAPS was the first application to be developed and was intended to reengineer the existing military personnel information system currently in use throughout the Air Force. Subsequent applications that were to be integrated as part of the initiative included revised supply, finance and accounting, medical, payroll, and base-level maintenance functions. The scope of the initiative was significant. In addition to the upgrade of the base-level business functions, the new applications were required to support a seamless interface at the headquarters level. Thus, almost all key Air Force information systems would be impacted in one way or another.

Lt. Col. Thompson noted several key features of the Air Force Business Process Modernization Initiative:

- **Client/Server architecture -** The existing mainframe computers and associated video display terminals were to be replaced by client/server architectures at each base and at each command headquarters.

- **Open systems -** The current dependence on vendor-specific, proprietary operating systems and database management systems was to be replaced by open system, standards-based architectures. A POSIX-compliant operating system had been selected as part of the software architecture for MAPS and the other Air Force information systems that were to be reengineered.

- **Standard data elements -** The efficient flow of data from one DoD information system to another was an important objective of the initiative. In order to achieve a high level of interoperability, the revised Air Force systems, including MAPS, had to adhere to a standard set of data definitions. The Defense Information Systems Agency (DISA) was responsible for control of the data standardization effort.

- **Process modeling -** All of the business processes that fell under the modernization initiative were to be modeled using the ICAM definition language (IDEF). This modeling effort was important to ensure the efficiency and interoperability of the various information systems that would be reengineered as part of the initiative.

- **Integrated databases -** An important aspect of the modernization initiative was the intent to move away from "stove-piped" business applications, each with its own database and unique application characteristics. Therefore, MAPS had to include an integrated database that could be accessed by the various user applications using a common data interface. The intent was for any given data element to be entered only once, at the point of origination. The data would then be made available to other applications. Development and control of the logical and physical data models rested with DISA, and again the MAPS design had to comply with higher-level requirements.

- **Maximum use of COTS software components -** The use of commercial software packages was strongly encouraged. As part of the modernization initiative, special waivers had to be obtained to develop unique software applications if a commercial counterpart that met the defined requirements was available.

- **Technical Architecture Framework for Information Management (TAFIM) -** All of the revised information systems that comprised the modernization initiative, including MAPS, were required to be designed and implemented in accordance with the DoD TAFIM. They were required to demonstrate Level-3 compliance with the Defense Information Infrastructure Common Operating Environment (DII COE).

1.3 Project Description

Lt. Col. Thompson's staff briefed him on the key project events and the technical and design characteristics of the MAPS project. MAPS began in the summer of 1996. It had been under development since that time by the Air Force's Administrative Systems Development Activity at Ridgway Air Force Base. All of the personnel involved in the MAPS development effort were organic to the activity. That is, they were either civilian or military personnel directly employed by the Air Force. The system and software requirements and high-level design were defined during the first year of the MAPS development. In November 1998, a briefing was given to the DoD MAISRC oversight group to support a Milestone III decision. Serious concerns were voiced by the members of the group during the briefing. The major issues focused on the development of MAPS and included the following:

- The original development schedule had been slipping on an incremental basis. The revised “get well” schedule presented by the previous Program Manager appeared to be unrealistic and could not be substantiated based upon the development performance to date.

- Similar to the schedule issue, there was no credible basis for the cost projections presented to the MAISRC. It appeared to the MAISRC that the cost of the software was driven by the number of development personnel available, not by the size and capability of the software that had to be developed.

The original MAPS development plan called for two incremental deliveries of the required capability. When Lt. Col. Thompson arrived at Ridgway in November 1998, the software for the first incremental release was under development.

MAPS began under a tailored MIL-STD-498 life-cycle process and was transitioning to IEEE/EIA 12207. The software development languages included both Ada 95 and C. Development tools included a state of the art Ada programming support environment, a graphical user interface (GUI) generator, and a report generator. A COTS relational database was also an integral part of the design.

The MAPS software design included twenty-four functionally defined Configuration Items (CIs). Thirteen of these were allocated to Increment 1 of the development and nine were allocated to Increment 2. The remaining two CIs were data conversion software. For each of these CIs, access to the database was to be implemented using SQL. User access and interface was designed to be implemented using predefined, “user friendly” screens. Site operators had additional access using SQL. The user interface was to be developed using X-Windows and was designed to be CDE compliant.

1.4 System Architecture and Functionality

The primary objective of the MAPS project was to reengineer the existing Air Force military personnel information system to add new functionality and to meet the overall integrated system requirements defined by the Business Process Modernization Initiative. To fully understand the technical implications of migrating the existing system to the new design, Lt. Col. Thompson compared the architecture and functionality of the current military personnel system with the MAPS requirements and specifications.

1.4.1 Current Personnel System

Figure 8B-1 shows the hardware architecture for the current personnel system. The current system actually consists of two separate information systems. One resides at the base level and the other at command headquarters. Both the base level and the headquarters implementations were based on the use of mainframe computers and video display terminals. The applications for both legacy systems were written in COBOL and included hierarchical databases. Both incorporated character-oriented, non-graphical user interfaces.

The operating concept of the current system includes periodic data transactions from the base-level system to the headquarters-level system. Selected data was uploaded to headquarters every 24 hours. As with many legacy information systems, the current military personnel implementation had experienced a significant number of problems with respect to inconsistent edits between the two systems. Part of this was attributable to the base-level system requiring very loose edits, while the headquarters system was much more constrained. Consequently, there was a large rejection rate for data that was uploaded to the headquarters system and data was often lost in the transaction process.

To access data at the base level from the headquarters database, users had to log in and connect to the system over standard phone lines. This access approach had proven to be unreliable and added to the problems associated with transferring data.

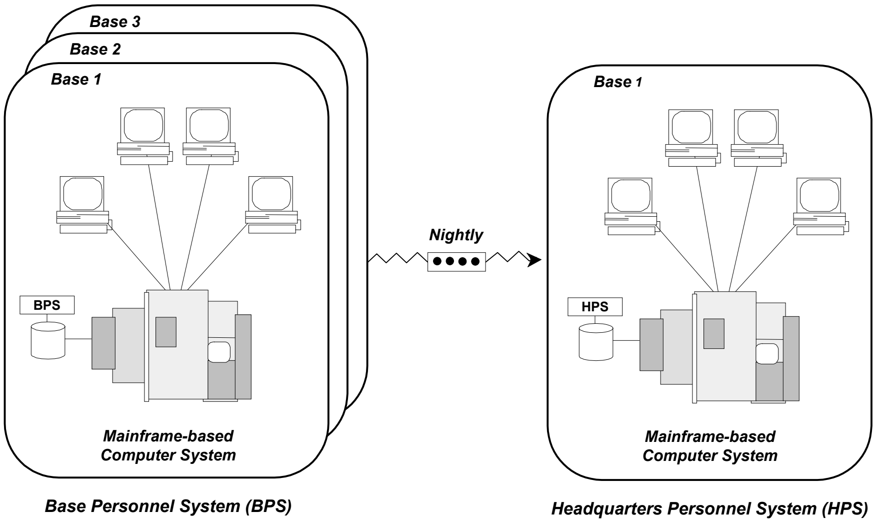

Base 3
Base 2
Base 1
BPS
Mainframe-based Computer System
Nightly
Base 1
HPS
Mainframe-based Computer System
Base Personnel System (BPS)
Headquarters Personnel System (HPS)

Figure 8B-1. Existing Air Force Military Personnel Information System Architecture

1.4.2 Military Automated Personnel System (MAPS)

The hardware architecture for MAPS is shown in Figure 8B-2. MAPS is designed as a single integrated personnel system that incorporates real-time data updates and access between the base- and headquarters-level systems. The headquarters portion of the system incorporates a mainframe computer that is used only for data storage. It is part of the headquarters local area network (LAN). MAPS incorporates a client/server design at both the base and headquarters levels. Data transfer between the levels is provided by a designated MILNET interface.

The MAPS client/server architecture integrates Graphical User Interface (GUI) and display functions on individual PCs, while the shared application functions reside on a UNIX-based server. This design is applicable at both the base and headquarters levels.

When MAPS is initially fielded at each Air Force base, it will be required to interface with the existing base-level information systems. These systems will gradually disappear as the Business Process Modernization Initiative progresses. As each existing information system is reengineered and integrated into the overall information system structure, all base-level applications will transition to a common enterprise architecture with access to a common database. As with MAPS, all interaction between applications will then occur through the shared database.

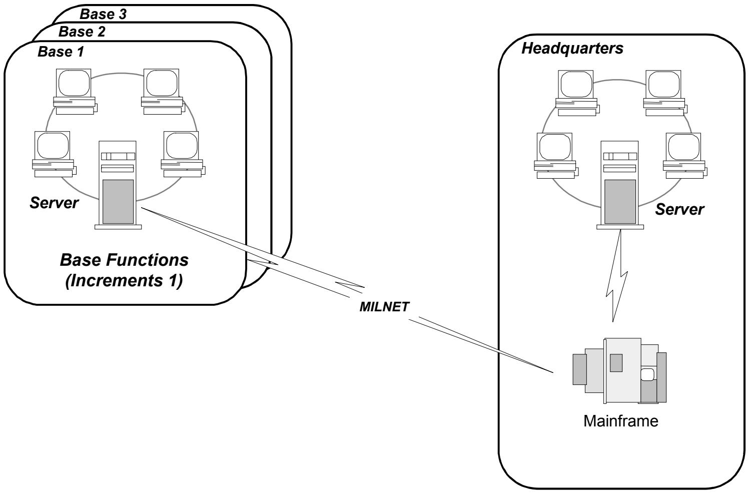


Figure 8B-2. MAPS System Architecture

The MAPS design incorporates two functional subsystems. As expected, these include the base-level functional subsystem and the headquarters-level functional subsystem. The base-level subsystem includes those standard functions that support the military personnel assigned to individual bases, or to commands, such as individual aircraft squadrons, that are resident on base. The type of personnel data that must be available from MAPS at the base level includes individual information on each officer and enlisted person assigned to the base. These data include age, rank, skill level, training history, individual personnel assignment and promotion history, and information pertinent to past performance evaluations. The base-level MAPS subsystem also contains personnel information at the command level, such as squadron mobilization personnel requirements, casualty data, skill profiles, and personnel replacement priority information.

The MAPS headquarters subsystem includes military personnel functions that generally support higher-level information requirements than those needed at the base level. The headquarters subsystem provides information that supports overall force mobilization, strategic planning, and analysis of force manpower requirements. For example, if a senior Air Force commander wants to deploy an offensive air superiority fighter such as the F15-E, the headquarters subsystem can provide information about the location of each F15-E squadron, and the availability and training history of the pilots, maintenance personnel, and other support crew. If the Air Force needed to plan for night air sorties into mountainous terrain, MAPS would help identify those squadrons with the appropriate qualifications.

The overall MAPS development plan called for the subsystems to be developed and delivered in separate increments. Increment 1 would include the base-level functions, and Increment 2 would include the

Headquarters functions. In addition to development of the respective increment functionality, MAPS required that the data from the current military personnel information system be converted and entered into the redesigned MAPS data structures. As such, the MAPS software development effort included the development of data conversion software for both the base-level and the headquarters-level databases.

2

Getting the Project Under Control

After his review of the MAPS development effort, Lt. Col. Thompson knew that he was facing a big challenge. A detailed review of the development and management processes revealed that the project was essentially managed with milestone schedules and viewgraphs. By mid-1998, the software development schedule milestones had begun to slip on a regular basis. Although this was evident in the milestone charts, no action was being taken to identify and correct the underlying causes. An analysis of the problem report data in the configuration management database showed that many more problem reports were being opened than were being closed. All of the available personnel were assigned to implementing and testing the code to meet the defined schedule for Increment 1. There wasn't enough time to keep up with the problem fixes at this stage of the development.

To gain control over the MAPS project, Lt. Col. Thompson had to address two key issues. The primary issue was the development schedule and progress. Lt. Col. Thompson had to assess the feasibility of the current schedule and determine why performance against the schedule was lagging. Second, he had to address the overall product quality of the developed products. Based upon past experience, Lt. Col. Thompson knew that the software defects represented in the open problem report backlog had a lot to do with the schedule pressures. The schedule had also limited the time spent resolving and closing problems. Given the increased visibility of the project after the results of the MAISRC review, Lt. Col. Thompson knew that the system had to work correctly when it was initially fielded.

By this time, it was clear to Lt. Col. Thompson that he needed better and more detailed information to manage the critical software issues. To help him get the information, he assigned one of the members of his project staff, Jennifer Cooper, as the MAPS measurement lead. Ms. Cooper had previous experience with implementing a measurement process, but this would be the first time she had to tailor and apply measurement for an existing project. Ms. Cooper met with Lt. Col. Thompson to identify and prioritize the major software issues to be addressed by the measurement effort.

From the discussion, it was clear that Lt. Col. Thompson would give the measurement activities a high priority. He intended to use the measurement results to help get the project back on track, and also to show senior management how the project was progressing.

Lt. Col. Thompson and Ms. Cooper discussed the potential problems related to implementing measurement on an existing program. Although all of the measurement data that they wanted would not be immediately available, they had enough basic information to start to address the key issues. They both decided that it would be a good idea to review the measurement results on a weekly basis.

As one major step in gaining control of the MAPS development, Lt. Col. Thompson put together an Integrated Product Team (IPT) consisting of representatives of a number of organizations associated with MAPS. These included the base-level and headquarters user communities, the designers, the test and integration organization, quality assurance, and installation personnel. Ms. Cooper was also a member of the IPT. The IPT's task was to identify and prioritize the risks to the project. The major risk they identified was in converting the existing databases to the shared relational database that would be accessed not only by MAPS but by future applications as well. Their concerns were two-fold. First, they were concerned that the existing data would be so error-prone that it would make the conversion process labor-intensive and would result in a schedule slippage. They estimated the probability of this occurring at 0.5. If this did occur, the impact would be to increase the effort as well as schedule. Without more information, they viewed their probability estimate as a guess more than anything else. They also could not come up with a precise impact estimate. Their second concern was that the process of data standardization needed to make the shared data concept a reality would get bogged down in organizational battles. They estimated the probability that this would happen as 0.70. There was currently high-level support within the Air Force and within DISA for data standardization. Their concern was that this might change with future personnel changes. They estimated the impact on MAPS as minor. The real impact would be on the Air Force vision of data sharing and interoperability.

The IPT met with Lt. Col. Thompson and recommended two risk mitigation strategies. To handle the error-prone data, they suggested that very close attention be paid to the first few data conversion efforts. The IPT felt that this would give them a much better sense of the extent of the problem and they could replan for more manual effort in the conversion phase if necessary. For the data standardization effort, they suggested that the MAPS project be proactive in working with other Air Force organizations and DISA to identify shared data and reach a consensus on the data model and data elements. They also identified a middleware package that could be used to translate data between MAPS and the other databases.

Lt. Col. Thompson gave the IPT the go-ahead to implement these recommendations. He asked Ms. Cooper to develop a means to quantify the extent of any problems related to data conversion. This quantitative data would be used as an objective basis to change the plan, if that proved to be necessary.

2.1 Evaluating the Project Management Plan

When Lt. Col. Thompson reviewed the MAPS development plan, he tried to identify how the original schedules and staffing requirements had been established. The most detailed schedule information available was in the form of Gantt charts showing major project milestones and dates. There was little detail with respect to the low-level MAPS software development activities and associated CI development tasks. There was a project Work Breakdown Structure (WBS), but it seemed to apply only loosely to the current tasks. It appeared that the overall development schedule was driven by the required delivery date of the system. Key development activities were scheduled very optimistically to meet the delivery date.

There was no MAPS staffing plan that allocated personnel resources to specific development tasks. A total of 40 software personnel were assigned full time to the MAPS project. All were available through the planned delivery date for Increment 2. The people were being applied to the project on a level-of-effort basis.

The first question Lt. Col. Thompson had to answer was whether or not the original MAPS project schedule was realistic, given the projected level of staffing and the overall performance of the development team to date.

Lt. Col. Thompson asked Ms. Cooper to generate an independent schedule estimate based upon the software size and the expected productivity. Although this sounded like a straightforward request, Ms. Cooper understood that the characteristics of the project required two separate sets of analysis. There were two different “types” of development taking place, each described by distinct development approaches. These included:

- Development of the application software for both incremental deliveries. This development effort was based on the use of a commercial database, SQL, Ada, and GUI-generation and report-generation tools.

- Development of the data conversion software. This development effort could best be described as a “typical” support software development effort using a high-order language with minimal process requirements.

Ms. Cooper needed to estimate the size of the software to be developed in order to generate a new estimate of the MAPS development schedule. She decided to use function points as the basic size measure for the Increment 1 and 2 application software because of the mix of languages (Ada, SQL, code generators). Ms.

Cooper used two methods to calculate the required productivity figures. In addition to a simple functional size to effort ratio, Ms. Cooper used a software cost model that accepted function points as a data input. The model also took into account the productivity impact of language type and reused code.

For the data conversion software, Ms. Cooper used lines of code to estimate the software size. In this case, lines of code seemed a better choice because she was not readily able to convert the sizing information to function points.

Ms. Cooper spent several weeks with the development team to arrive at the function point counts and the lines-of-code estimates. The function point counts were based upon the methodology defined in the Function Point Counting Practices Manual from the International Function Point Users Group (IFPUG). The responsible team leaders generated estimates of source lines of code for each of the application functions. Ms. Cooper summarized the sizing results for Lt. Col. Thompson as shown in Figure 8B-3.

The figure shows the estimated size for each of the CIs in Increments 1 and 2. The figure also shows the primary language and the projected number of low-level design components or units.

The relational database and the Ada to SQL bindings inherent in the MAPS design were relatively new COTS software products. Input screens and reports were generated by 4GLs.

Ms. Cooper's projections indicated the following:

- The minimum schedule to develop both functional increments would be four months longer than the planned development schedule.

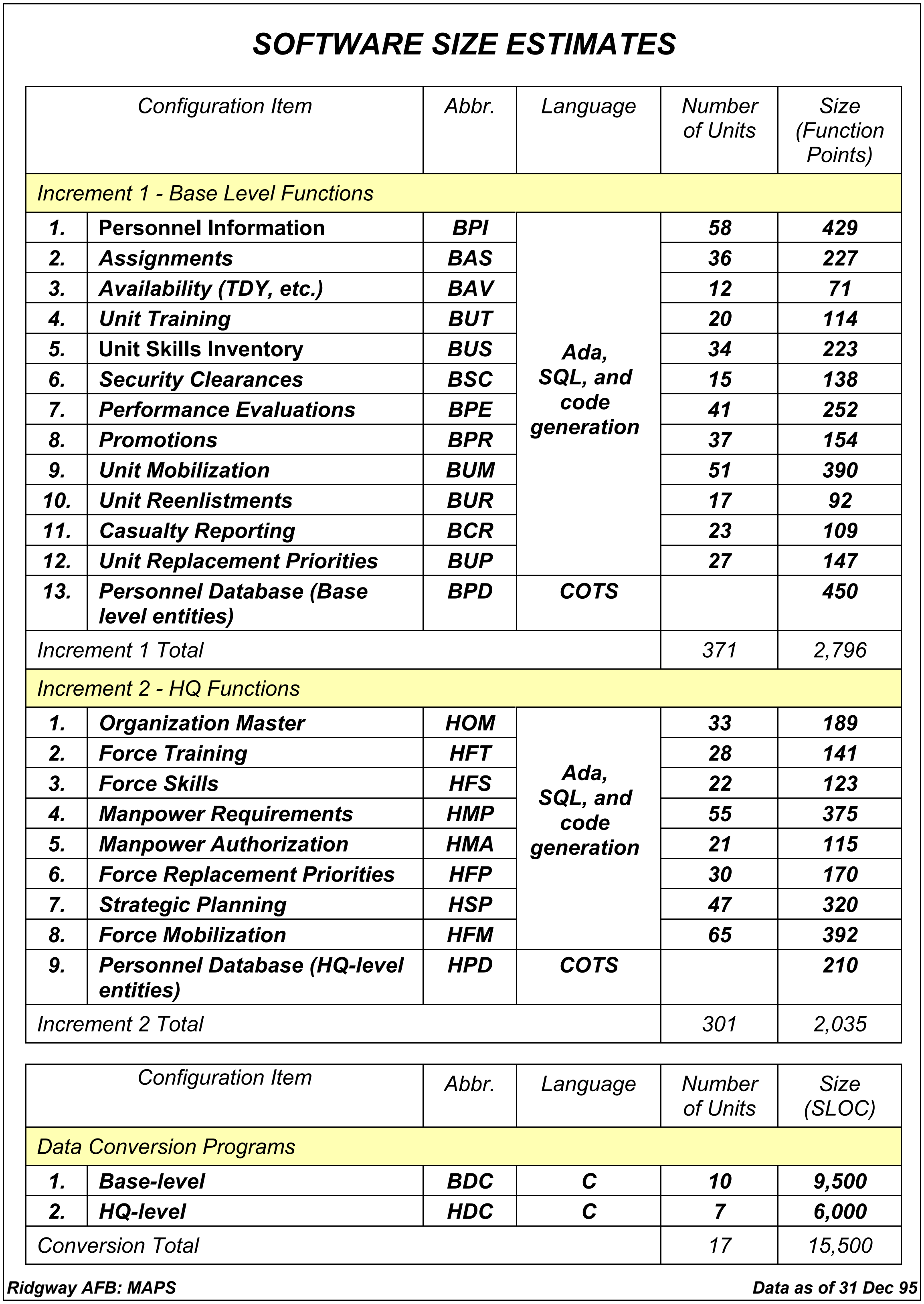
- In order to meet even the extended schedule, the MAPS development staffing levels would have to be significantly increased.

Although these analysis results were expected, they indicated that Lt. Col. Thompson would have to replan the remainder of the MAPS project to define a more realistic development plan.

2.2 Revising the Project Management Plan

Lt. Col. Thompson used the cost model estimates as the basis for a revised project management plan. He asked Ms. Cooper to show the new schedule in the form of a Gantt chart. This revised schedule is shown in Figure 8B-4.

The revised schedule began with the completed activities. The system requirements and high-level design activities were ongoing from July 1997 through May 1998.



SOFTWARE SIZE ESTIMATES

Configuration Item		Abbr.	Language	Number of Units	Size (Function Points)
Increment 1 - Base Level Functions					
1.	Personnel Information	BPI	Ada, SQL, and code generation	58	429
2.	Assignments	BAS		36	227
3.	Availability (TDY, etc.)	BAV		12	71
4.	Unit Training	BUT		20	114
5.	Unit Skills Inventory	BUS		34	223
6.	Security Clearances	BSC		15	138
7.	Performance Evaluations	BPE		41	252
8.	Promotions	BPR		37	154
9.	Unit Mobilization	BUM		51	390
10.	Unit Reenlistments	BUR		17	92
11.	Casualty Reporting	BCR		23	109
12.	Unit Replacement Priorities	BUP		27	147
13.	Personnel Database (Base level entities)	BPD	COTS		450
Increment 1 Total				371	2,796
Increment 2 - HQ Functions					
1.	Organization Master	HOM	Ada, SQL, and code generation	33	189
2.	Force Training	HFT		28	141
3.	Force Skills	HFS		22	123
4.	Manpower Requirements	HMP		55	375
5.	Manpower Authorization	HMA		21	115
6.	Force Replacement Priorities	HFP		30	170
7.	Strategic Planning	HSP		47	320
8.	Force Mobilization	HFM		65	392
9.	Personnel Database (HQ-level entities)	HPD	COTS		210
Increment 2 Total				301	2,035

Configuration Item		Abbr.	Language	Number of Units	Size (SLOC)
Data Conversion Programs					
1.	Base-level	BDC	C	10	9,500
2.	HQ-level	HDC	C	7	6,000
Conversion Total				17	15,500

Ridgway AFB: MAPS

Data as of 31 Dec 95

Figure 8B-3. The MAPS software size estimates were entered into a cost model

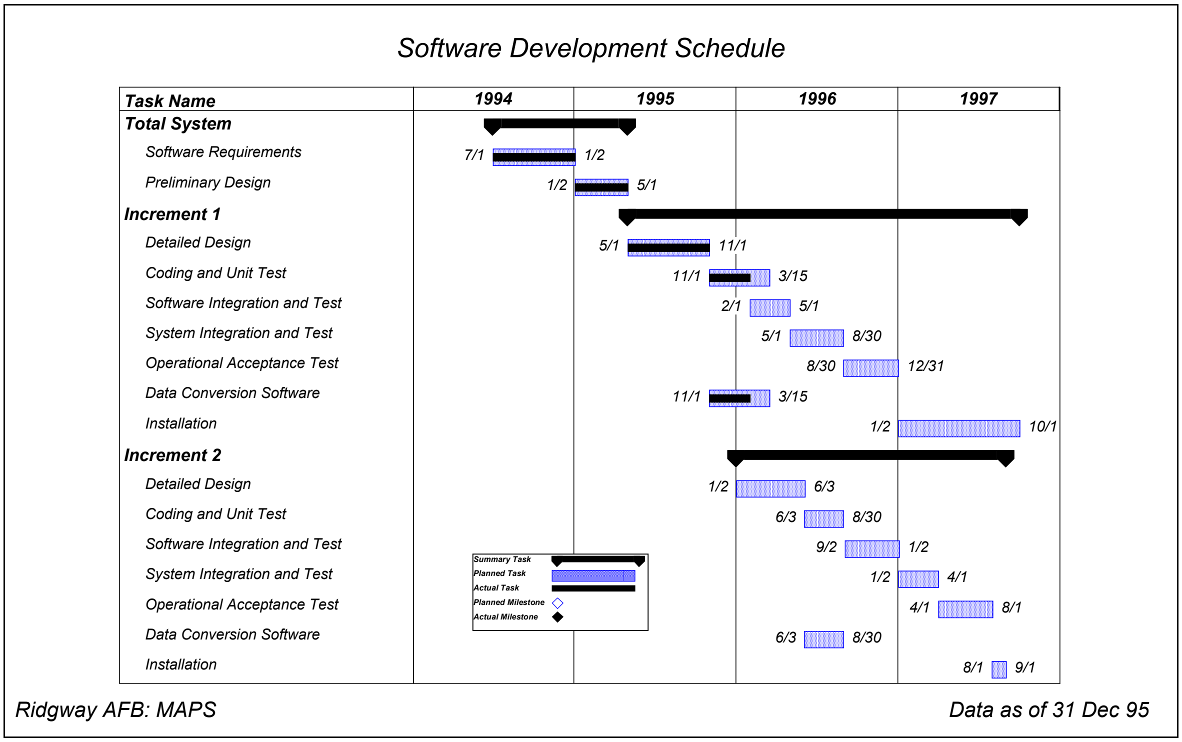


Figure 8B-4. The MAPS development staffing levels would have to be significantly increased to meet the extended schedule

Top-level requirements and design were completed early in the development effort for the entire system. With these activities complete, the revised schedule called for the independent development of the application software in two parallel increments as previously defined. The development of each increment included detailed design, coding, and integration and test.

The detailed design for Increment 1 was completed in November of 1998. Increment 1 was to be fielded by the end of 2000. Detailed design for Increment 2 was scheduled to begin in early 1999. Increment 2 was scheduled for delivery in mid 2000. The data conversion software was scheduled for parallel development with the respective functional increments. Data conversion and installation was scheduled to occur over a ten-month period for Increment 1 and a one-month period for Increment 2.

Lt. Col. Thompson identified two major development activities on the critical path. These were the Personnel Information CI for the base-level subsystem and the data conversion software for both functional increments. The “Personnel Information” CI was critical because it had to be completed before the other CIs could be integrated and tested. The data conversion software was critical because it was needed to convert the existing databases at each base and at headquarters. The data conversion effort had already been identified as a high-risk item by the IPT. The data conversion software had to be completed, and had to work properly, before the MAPS increments could be fielded. Lt. Col. Thompson decided to track these critical-path items closely.

The results of the productivity analysis were also used as the basis for the revised MAPS staffing plan. The projected effort allocations for Increment 1 and Increment 2 were graphed as shown in Figure 8B-5. When Lt. Col. Thompson reviewed the incremental effort allocation, he noted that the peak, full-time staffing requirement did not exceed 35 people. Since the schedule called for the MAPS increments to be developed in parallel, Lt. Col. Thompson asked Ms. Cooper to generate a system-level effort allocation graph. This graph is depicted in Figure 8B-6.

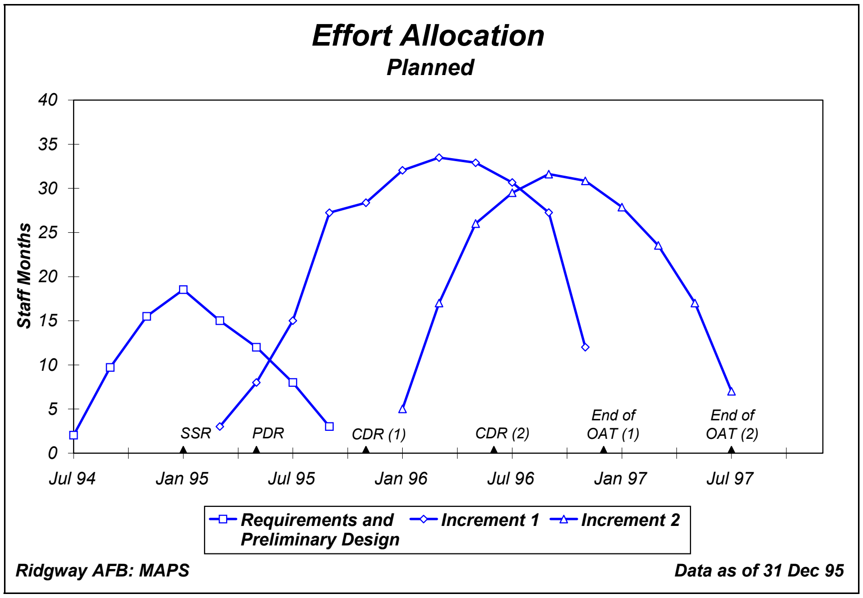


Figure 8B-5. The number of people currently assigned to the MAPS development team was not adequate to meet the peak staffing requirements

When Lt. Col. Thompson looked at the total system effort profile that aggregated the individual effort requirements, several things became apparent. It was clear that the number of people currently assigned to the development team was not adequate to meet the peak staffing requirements that would occur in 1999. Even more important, the level staffing profile of 40 people did not meet the needs of the project. The development had been inefficiently overstaffed through 1998 and was then projected to experience shortfalls in 1999.

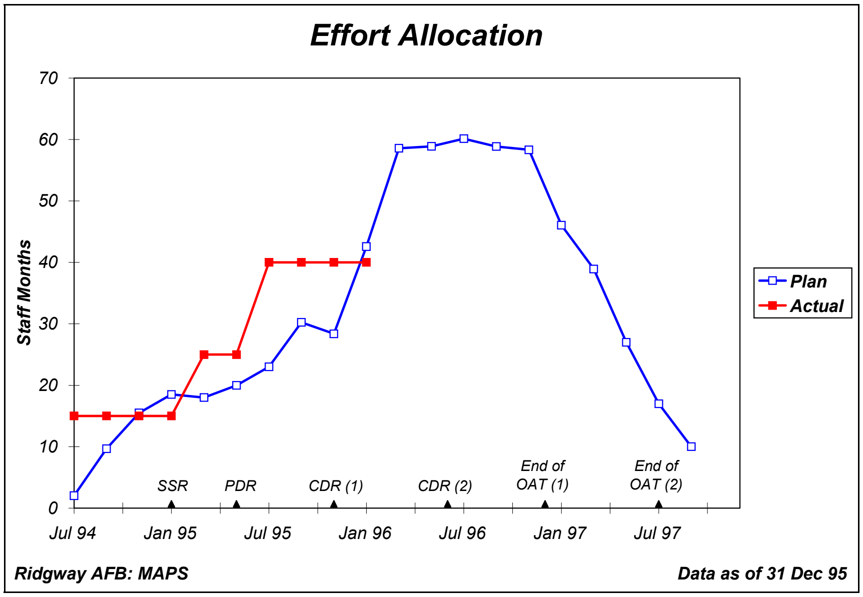


Figure 8B-6. The effort allocation indicator showed that additional funding must be allocated to meet the 1996 staffing requirements

Lt. Col. Thompson used the measurement results to brief senior management about some of the issues impacting the development of MAPS. They agreed with his overall assessment and added four months to the development schedule. They also agreed to allocate additional funding to support the 1999 staffing requirements. The plan was to use qualified Air Force personnel from other projects and to hire outside contractors to help with detailed design, coding, and software integration and test for the MAPS Increment 2 development.

2.3 Tracking Performance Against the Revised Plan

Once the new schedule and staffing plans were in place, Lt. Col. Thompson’s concerns shifted from evaluating the feasibility of the plans to assessing performance against the plans. Although the milestone data continued to be useful in addressing the schedule and progress issues, more detailed information was required to track the degree of completion of the key development activities and products. The need for this information was clear as Lt. Col. Thompson reviewed the Gantt chart that represented the revised project schedule (Figure 8B-4). The milestone schedule indicated that detailed design for Increment 1 had been completed and software implementation was well underway. Based on the schedule, about two-thirds of the time allocated for coding had already elapsed. This didn’t mean however, that two-thirds of the Increment 1 software had been coded. To get information about the degree of activity and product completion that they needed, Lt. Col. Thompson and Jennifer Cooper decided to implement several work unit progress measures.

Work unit progress measures compare the actual completion of associated work units for software products and activities against a pre-established plan. If objective completion criteria for each type of work unit are defined and adhered to, work unit progress measures provide for a clear determination of development progress. For each of the MAPS CI's, Ms. Cooper recommended that the project use counts of the number of design units implemented as the work unit progress measure. The design units represented the lowest practical level of measurement, and the data could easily be collected from the configuration management system. In this case, an implemented design unit was defined as passing unit test and being entered into the project library.

To generate the CI work unit progress indicators, Ms. Cooper first defined the planned rate of unit completion. Without detailed planning data available, Ms. Cooper generated a straight-line completion plan beginning with CDR and ending with the scheduled completion of the Increment 1 coding activity. In Ms. Cooper's previous experience with work unit progress measures, she had found that the more accurate plans for the cumulative number of work units completed over time often looked more like an S-shaped curve than a straight line. This was due to the fact that the first few units tended to be completed slowly, followed by a faster rate of completion as the activity progressed. Nearing the end of the software activity, the completion rates tended to slow again as the more difficult units tended to be completed last. For the MAPS work unit progress measures, the straight-line plan was not perfect, but was seen as a useful approximation. Everyone understood that they would not be too alarmed if progress lagged behind the straight-line plan at the beginning of the development activity.

Once Ms. Cooper had established the plan, she accessed the configuration management library to obtain a count of units completed to date. Specifically, she counted the number of units that had been entered into the library each week over the course of Increment 1 implementation. The resulting graph is shown in Figure 8B-7. The graph indicated that the CI implementation was progressing in accordance with the revised development plan.

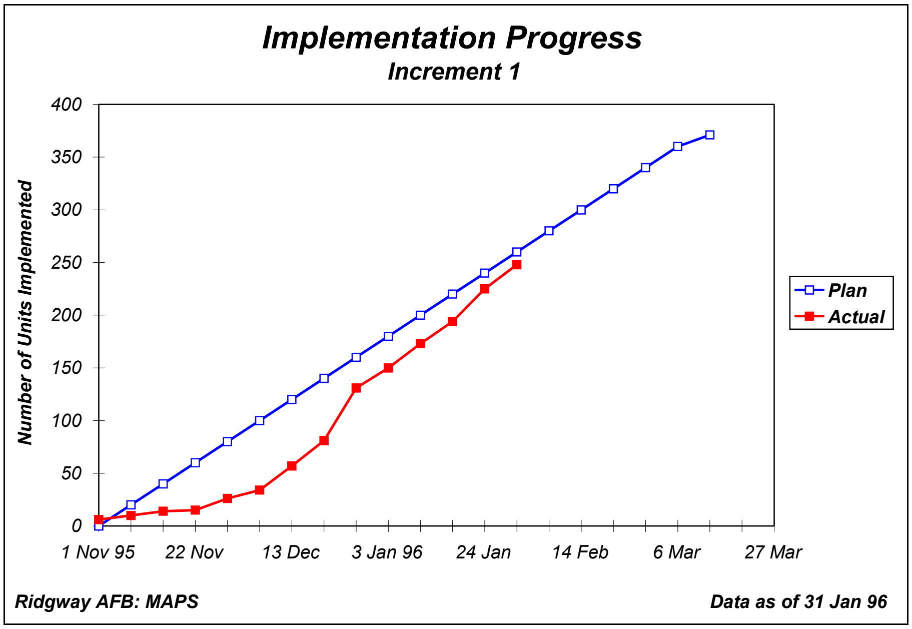


Figure 8B-7. The CI implementation progress indicator showed that current progress was in accordance with the revised development plan

Ms. Cooper knew that Lt. Col. Thompson wanted to emphasize software measures related to the schedule and progress issue. As such, she decided to track progress for the two items on the critical path very closely. These were the development of the Personnel Information CI and the development of the data conversion software. The Personnel Information CI was scheduled to be completed by March 1999. Ms. Cooper constructed a plan to track work unit progress for the single CI the same way she did it for the aggregate of the CIs in Increment 1. Again, the plan was derived by drawing a straight line between CDR and the scheduled end of the coding activity. The resulting indicator was graphed and is depicted in Figure 8B-8. When the actual number of design units were compared to the plan, it became immediately clear that progress on this critical CI was lagging significantly.

Ms. Cooper then decided to try to identify the source of the progress problem in the Personnel Information CI. She defined two new work unit progress indicators using a somewhat different perspective. She graphed the development progress data for the screens and reports separately from the units that performed internal processing. The screens and reports were being implemented using a 4GL while the internal processing code was being written in Ada.

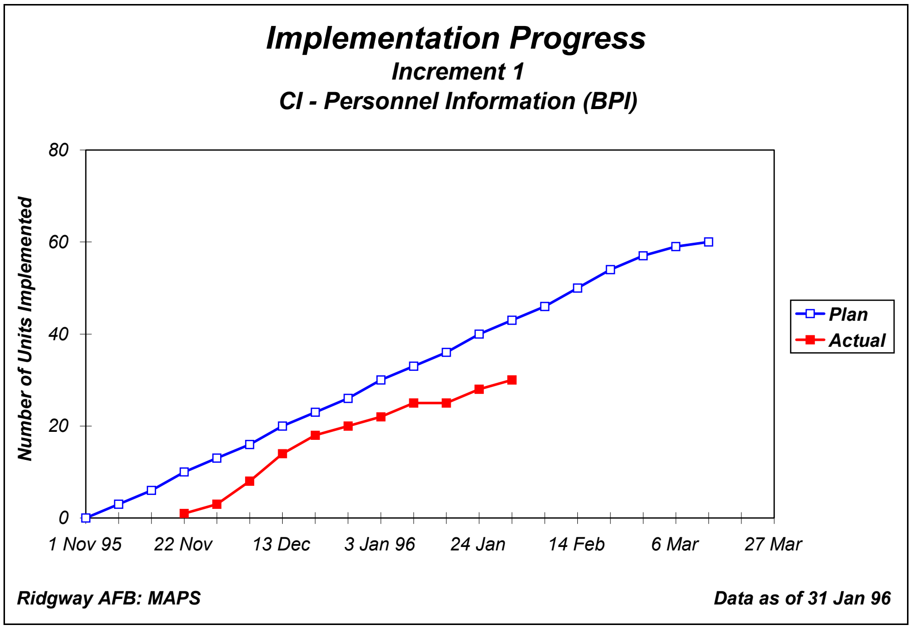


Figure 8B-8. An indicator that compared the actual number of design units to the plan showed the Personnel Information CI to be lagging significantly

The results are shown respectively in Figures 8B-9 and 8B-10 The measurement data showed that the screen and report development was on track and indicated that the problem was confined to the Ada code. When Lt. Col. Thompson investigated, he found out that the Ada developers were having difficulty interfacing their respective CIs to the COTS relational database. The problem was not critical from a technical perspective, but the workarounds were taking quite a bit of time to implement using SQL. Lt. Col. Thompson did several things to correct the interface problems. The first thing that he did was to bring in representatives from the COTS vendors to work on-site with the Ada developers to provide real-time support in resolving interface problems. Secondly, he had the development team conduct a one-time in-depth inspection of the CI's design and completed code. This inspection identified some design structures that were inefficient but could be corrected. Col. Thompson also assigned several of his most experienced Ada programmers to work on the Personnel Information CI in an attempt to correct the problem.

The other portion of Increment 1 that was on the critical path was the data conversion software for the base-level databases. In tracking work unit progress for this software, Ms. Cooper decided to count the lines of code that had been entered into the configuration management library, rather than counting the number of completed units.

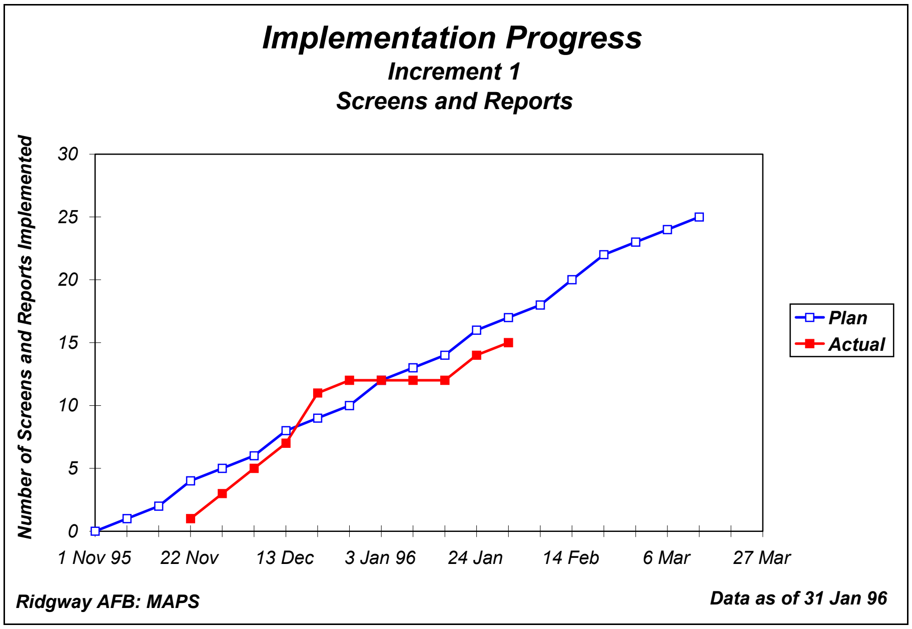


Figure 8B-9. The development progress indicator for screens and reports showed no problem

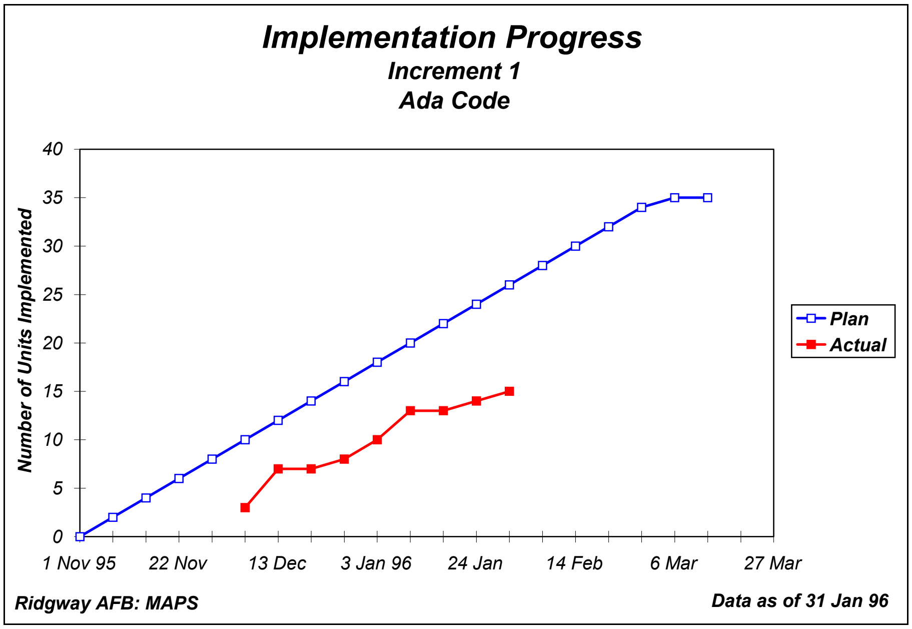


Figure 8B-10. The development progress indicator showed a significant problem in production of Ada code

She decided that completed lines of code was a better measure of progress than a count of units because the data conversion software was divided up into relatively few units and they varied drastically in size. The units were not equivalent and using them to track progress would have been misleading. Ms. Cooper generated the plan and actuals for the data conversion software and graphed the indicator as shown in Figure 8B-11.

The results showed that the data-conversion software development progress was on track.

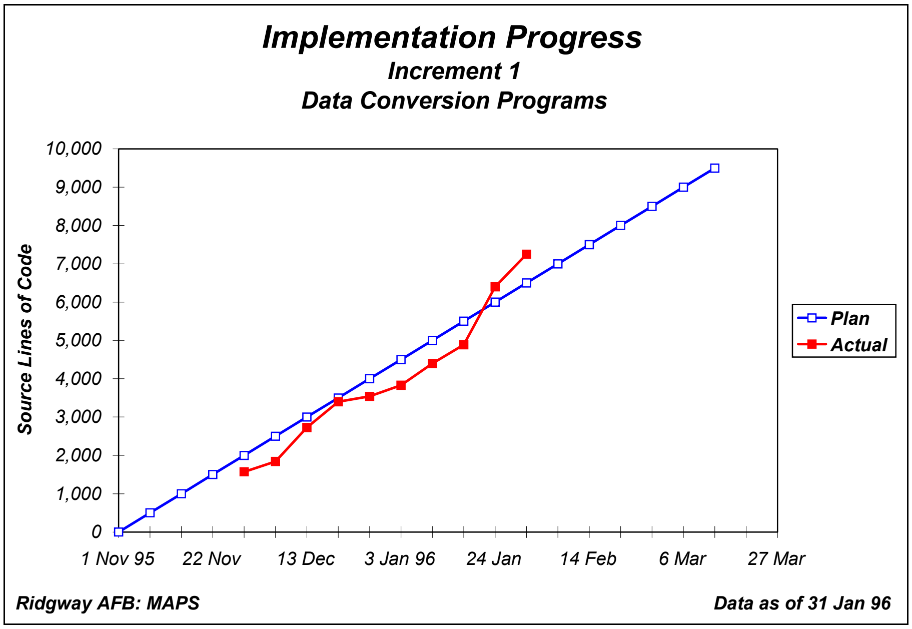


Figure 8B-11. The implementation progress indicator for data conversion showed no problem

3

Evaluating Readiness for Test

During 1999, the MAPS measurement process was effective in helping to manage the software development effort. Progress against the revised plan was sufficient enough to allow for the resolution of the problem reports that were previously backlogged. Additional personnel that were earlier added to the development team allowed for the concurrent development of both the base- and headquarters-level MAPS increments. The progress measures showed that Increment 1 was nearing the completion of integration and test, and some system level testing had already been conducted. The primary issue had shifted from schedule and progress to the quality of the software. The key concern was the readiness of the software for Operational Test and Evaluation.

3.1 Increment 1

As the initial 2000 delivery dates grew closer, Lt. Col. Thompson wanted to know if Increment 1 was ready to begin Operational Test. To help answer this question, Ms. Cooper defined a set of related indicators and graphed them as shown in Figure 8B-12a-d.

When Ms. Cooper first joined the MAPS project, the project had not been collecting effort data at the level of detail required to show how much effort was being applied to software rework. As an organic development activity, it was difficult to get the staff to record on their timecards how they actually applied their effort during the week. Since the emphasis had been on generating new code to meet the existing schedule, the development team didn’t see a need for the information anyway. As such, only development effort was collected as part of the time-reporting system. To get the data that she needed, Ms. Cooper asked one of the programmers to modify the problem reporting system to collect the “re-development” and “retesting” effort data related to software rework on a problem-by-problem basis.

The change in the process was briefed to the suppliers, and Ms. Cooper began to collect the data she needed to compare the amount of effort spent in rework vs. new development. The data was graphed and is presented in the lower right chart in Figure 8B-12d.

Readiness for Test
Increment 1

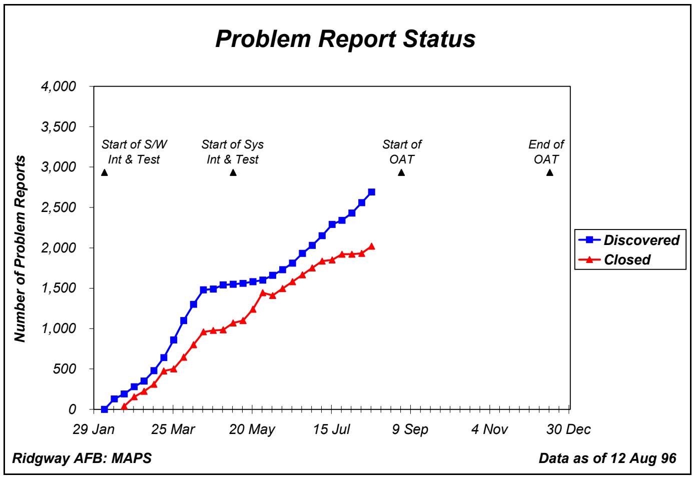


Figure 8B-12a

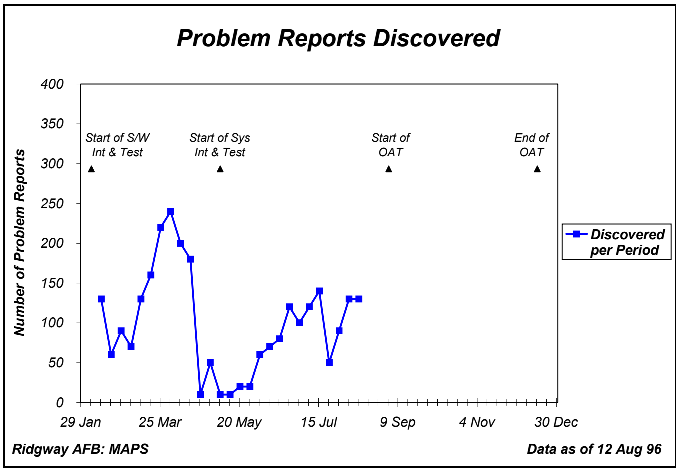


Figure 8B-12b

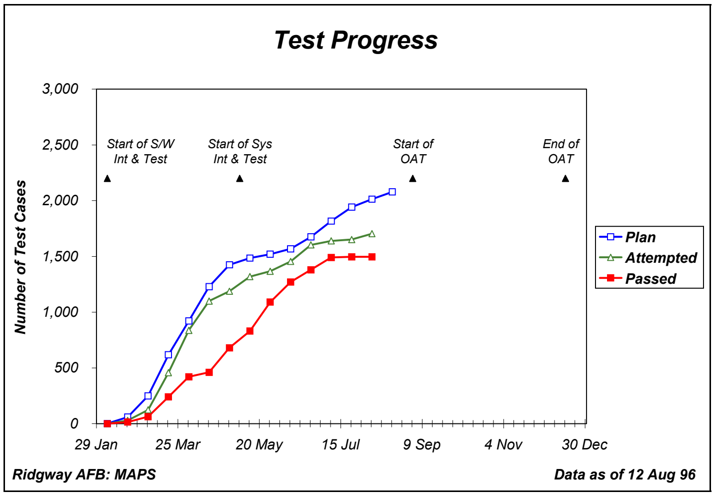


Figure 8B-12c

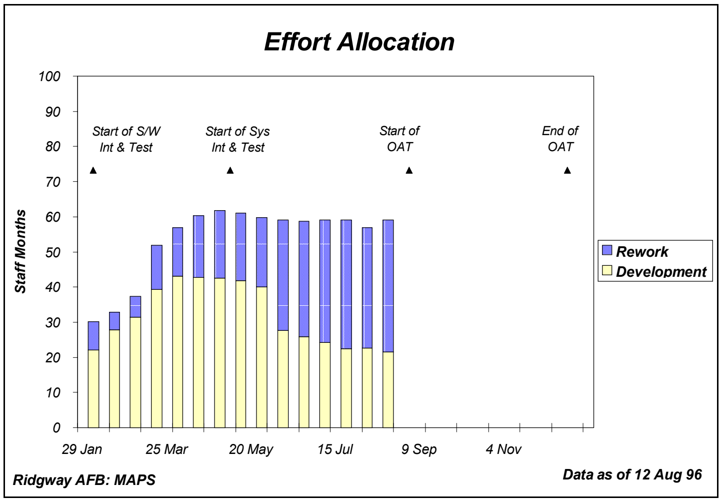


Figure 8B-12d

Ms. Cooper combined the rework effort data (Figure 8B-12d) with a work unit progress graph for cumulative problem reports (Figure 8B-12a) and a graph of the number of problem reports being opened on a weekly basis (Figure 8B-12b). She also included a graph of test case progress (Figure 8B-12c). This combination of indicators suggested that Increment 1 was not yet ready to begin Operational Test. Lt. Col. Thompson wanted to 1) see the open and closed problem report trends converging; 2) the number of new problems being discovered declining; 3) the number of test cases passed equal to the number planned; and 4) the amount of effort being applied for rework decreasing. The results indicated that the development staff was spending an increasing amount of time correcting new Increment 1 problems. This was a concern, because they should have been transitioning to the development of code for Increment 2. Lt. Col. Thompson met with Ms. Cooper and asked her for more information in order to identify what needed to be done to improve the situation. Specifically, he wanted information about the types of problems that were being reported. He was hoping that there was a common type of problem that could be effectively managed.

Ms. Cooper spent the better part of a week with several of the testing personnel reviewing the problem reports and classifying them as being related to performance, logic, interfaces, or other. She decided to implement this classification scheme as a permanent part of the problem reporting system so that the information would be readily available to support future analysis. The results of the classification effort were graphed and are depicted in Figure 8B-13. By far, the greatest number of the Increment 1 problems were related to performance deficiencies.

Ms. Cooper further classified the performance problems according to their sources. The results are shown in Figure 8B-14. The most common type of performance problem was due to the incorrect use of SQL by the suppliers.

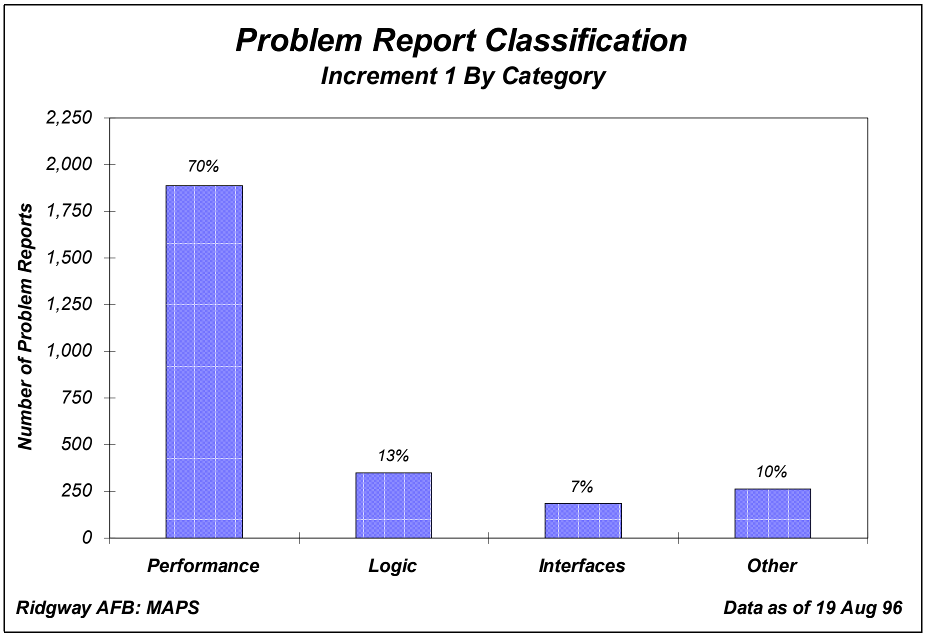


Figure 8B-13. The problem report classification indicator showed that the Increment 1 problems were related to performance deficiencies

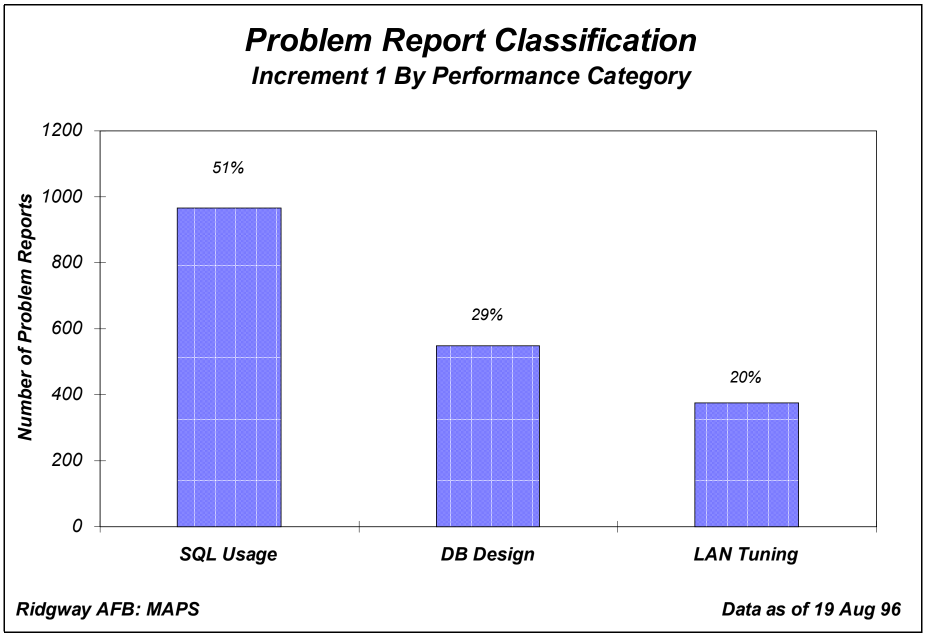


Figure 8B-14. The problem report classification indicator showed the cause of performance problems was the incorrect use of SQL by the suppliers

Ms. Cooper discussed the results of her analysis with Lt. Col. Thompson and pointed out that the MAPS development represented the first time that many of the people on the development team had used a relational database and SQL. The staff's previous experience had been with hierarchical databases and COBOL. This probably should not have been a surprise since the SQL issue was part of the reason for the previous Personnel Information CI development problems. Lt. Col. Thompson again decided to bring in some additional expertise to address the SQL issue. Although it wasn't the best approach this late in the project, the problems needed to be fixed quickly.

3.2 Increment 2

Increment 2 was scheduled for delivery early in 2000. According to the development schedule, Increment 2 should have been nearing the completion of system test by the end of February 2000. To assess the Increment 2 readiness for test status, Ms. Cooper generated the same combination of graphs using the same indicators as she had done for Increment 1. The results are shown in Figure 8B-15a-d.

This time the situation was much more encouraging. The trends for open and closed problem reports were converging, the discovery rate for new problems was declining rapidly, and the amount of rework was relatively low and stable. In addition, a comparison between the number of test cases planned, executed, and passed provided further evidence that testing was being completed in accordance with the schedule. Ms. Cooper wondered why the number of newly discovered problems was declining so rapidly. Was the software that much better? Were discovered problems not being reported? Had the testing stopped? The test progress results helped Ms. Cooper answer part of her question. Since testing was proceeding as scheduled, the lower number of new problem reports were not a result of reduced testing efforts. Ms. Cooper looked into the reporting process and found that the identified problems were still being consistently documented.

Ms. Cooper continued to track the classes of reported problems, as shown in Figure 8B-16. In contrast to the results for Increment 1, which had a high proportion of problems related to performance, the problems for Increment 2 were much more evenly distributed. The measurement data for Increment 2 indicated that the issues and problems that were experienced in Increment 1 had been successfully addressed. Lt. Col. Thompson's decisions had helped to focus the right resources where they were needed.

Readiness for Test

Increment 2

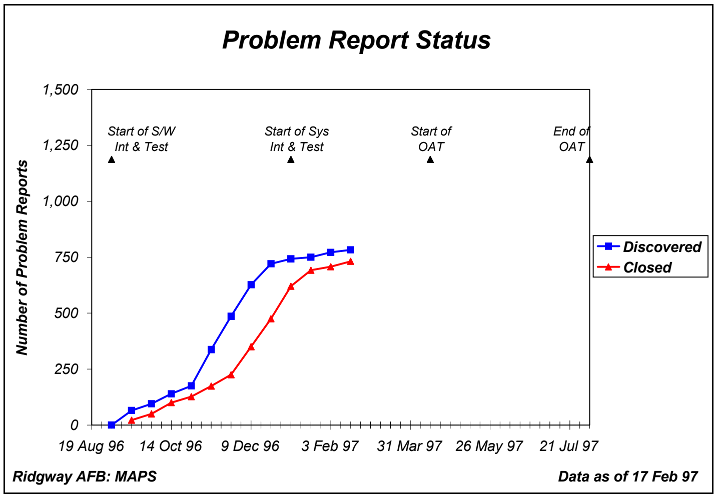


Figure 8B-15a

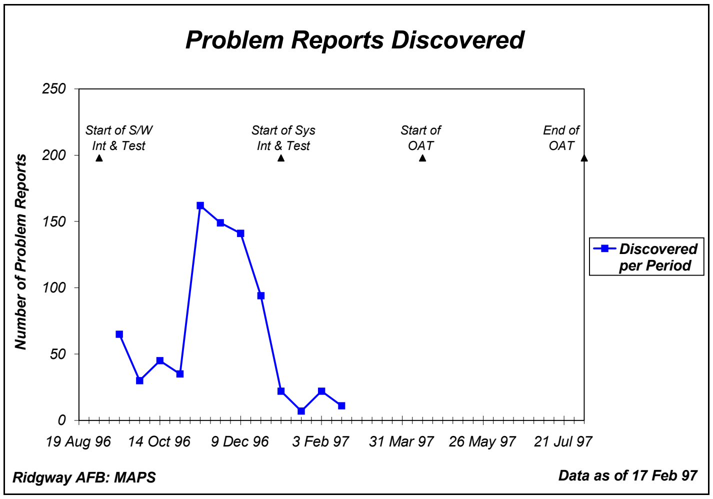


Figure 8B-15b

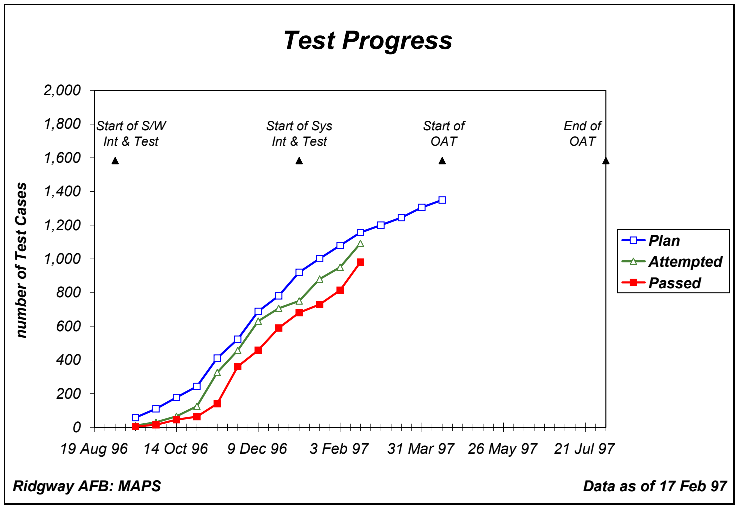


Figure 8B-15c

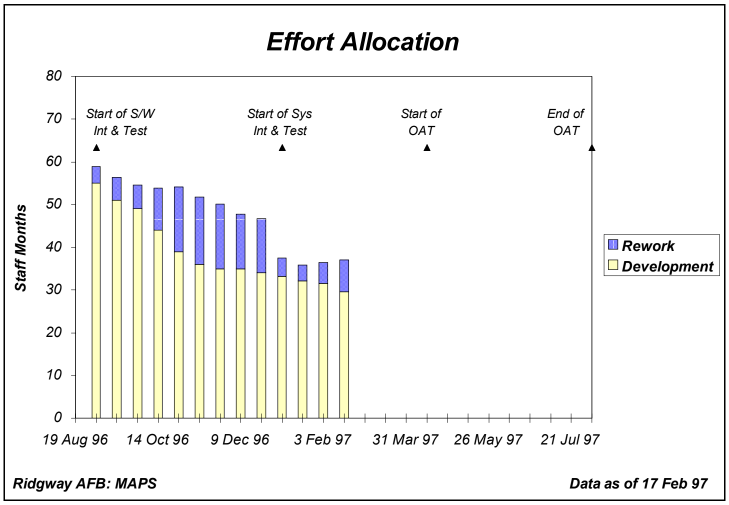


Figure 8B-15d

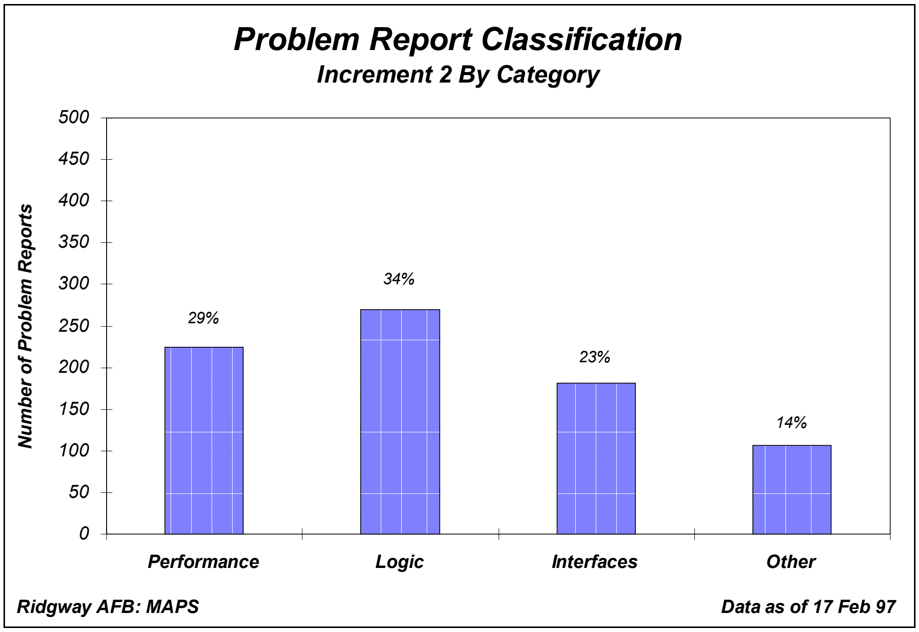


Figure 8B-16. The problem report classification indicator verified that the issues and problems experienced in Increment 1 had been successfully addressed

4

Installation and Software Support

With the development of MAPS proceeding according to plan, Lt. Col. Thompson asked Ms. Cooper to extend the measurement process to the fielding of the Increment 1 base-level systems. This was scheduled to occur throughout 2000, from January through October, with delivery of the systems occurring at a relatively constant rate.

To support the installation process, a total of ten people were assigned and divided into five teams. Each team was scheduled to spend two weeks installing MAPS at each of the 100 base-level sites. The work during the two-week installation period included data conversion, software installation, user training, and user support. After installation, the MAPS development team would provide support via a 24-hour help line. The plan called for each site to run the existing military personnel system concurrently with the newly installed MAPS for one week before shutting down the old system completely. The 100 base-level sites included all Air Force bases in the United States and overseas, Air Force Reserve commands, and selected Air National Guard units.

4.1 Increment 1 Installation

To track the installation progress, Ms. Cooper defined and graphed a simple work unit progress indicator as depicted in Figure 8B-17. Since data conversion was one of the major risks identified by the IPT, she wanted to have the earliest possible warning of any problems.

It is clear from the graph that the installations were behind schedule almost from the start. Ms. Cooper investigated and contacted each of the installation teams to identify the causes for the delays. She heard a consistent story. The old base-level system that MAPS was replacing had very loose edit requirements. It would accept almost any personnel data that was entered. The result was that the data conversion software that was written to the MAPS data specifications kept rejecting data that was in a different format from what was expected. This was not an easy problem to fix because each of the existing base-level databases was different from the others.

Ms. Cooper showed Lt. Col. Thompson a linear extrapolation of the actual installation data points. This is shown in Figure 8B-18. Based on the actual rate of progress, a total of fifteen months would be required to complete the installations, not ten months as originally planned. The rate of base installation was limited by the availability of teams. Based on the projection, Lt. Col. Thompson decided to extend the installation schedule. He also asked Ms. Cooper to provide an update to the projection as more data became available.

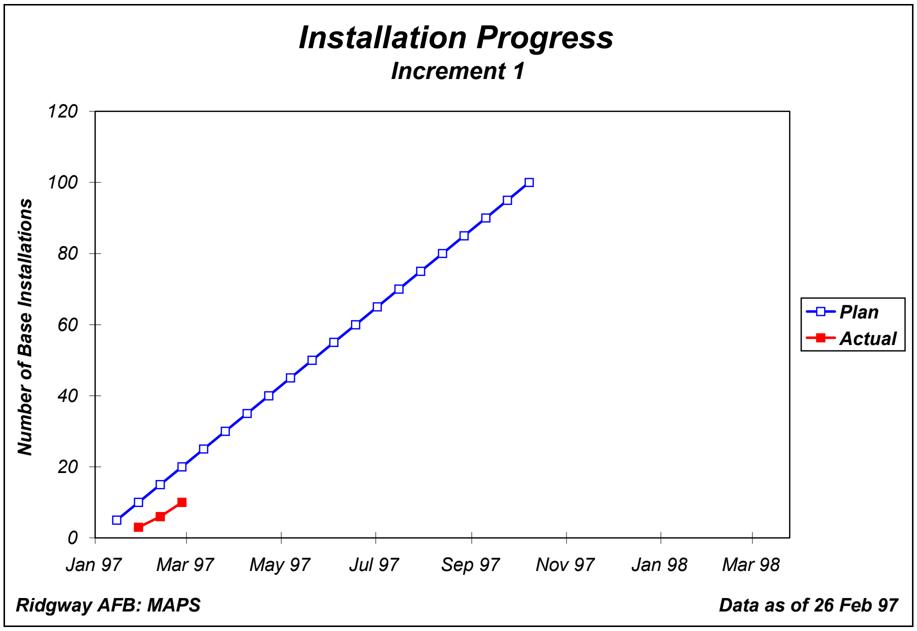


Figure 8B-17. A work unit progress indicator was used to monitor installation

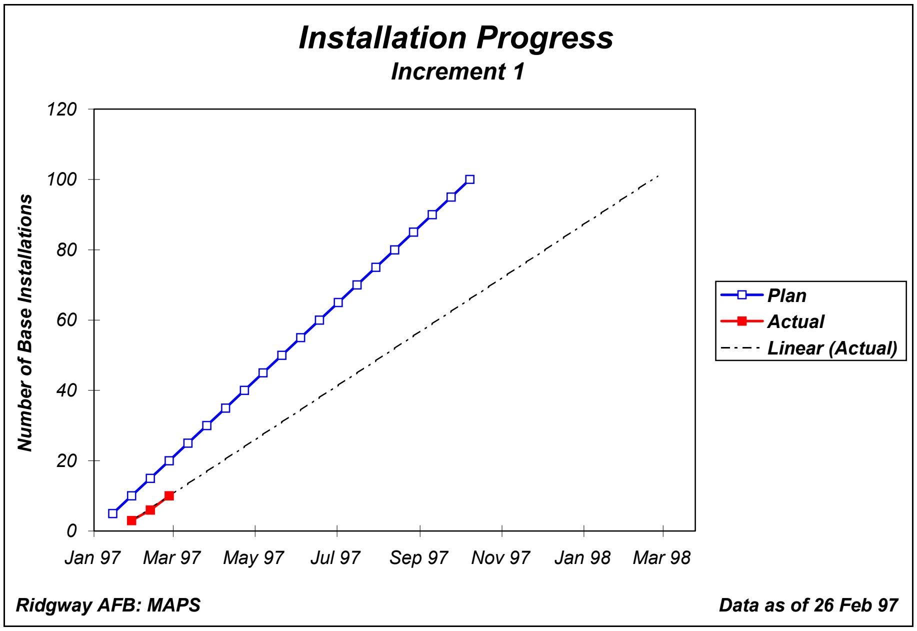


Figure 8B-18. An installation progress indicator provided early warning of an inadequate schedule

4.2 Software Support

By November 2000, MAPS had been installed at sixty-eight of the 100 base-level sites. As part of the measurement process, Ms. Cooper had been tracking and categorizing problem reports from the field. Given the previous problems on the project, it was important to Lt. Col. Thompson to address the users' concerns.

At the highest level, Ms. Cooper classified the problem reports as being related to hardware, software, or user error. She analyzed the software-related problem reports in more detail by focusing on those that were the result of defects in the design or the code. She classified the problems as related to performance, logic, interfaces with other systems, and other. The data coming in from the field showed that the most frequent type of problem was related to logic defects. This is shown in Figure 8B-19.

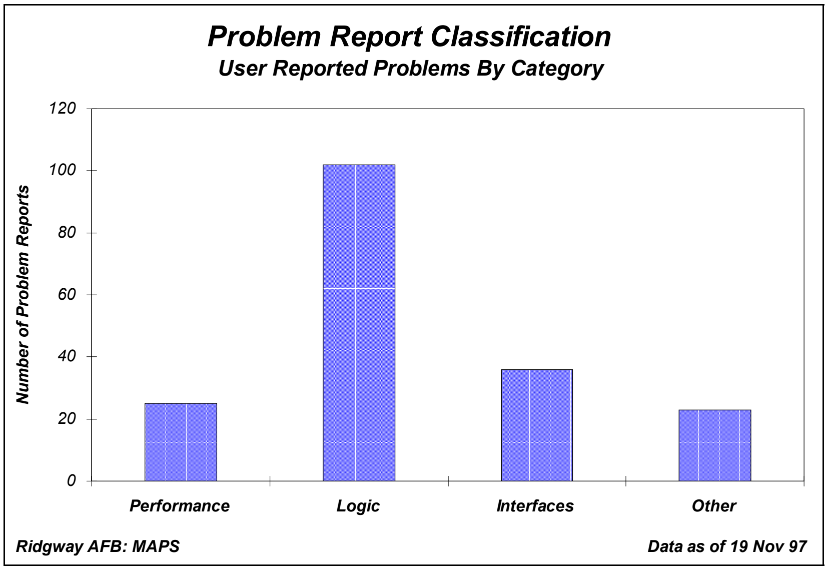


Figure 8B-19. Measurement data from the field showed that the most frequent problems were related to logic defects

Ms. Cooper also decided to classify the problems according to their source by identifying the CI that had to be changed in order to correct the problem. She graphed the ratio of problem reports to function points for each CI. The results were graphed as shown in Figure 8B-20. Ms. Cooper found that the Unit Mobilization (BUM) CI accounted for a disproportionate number of defects. Clearly there was a problem with this particular CI.

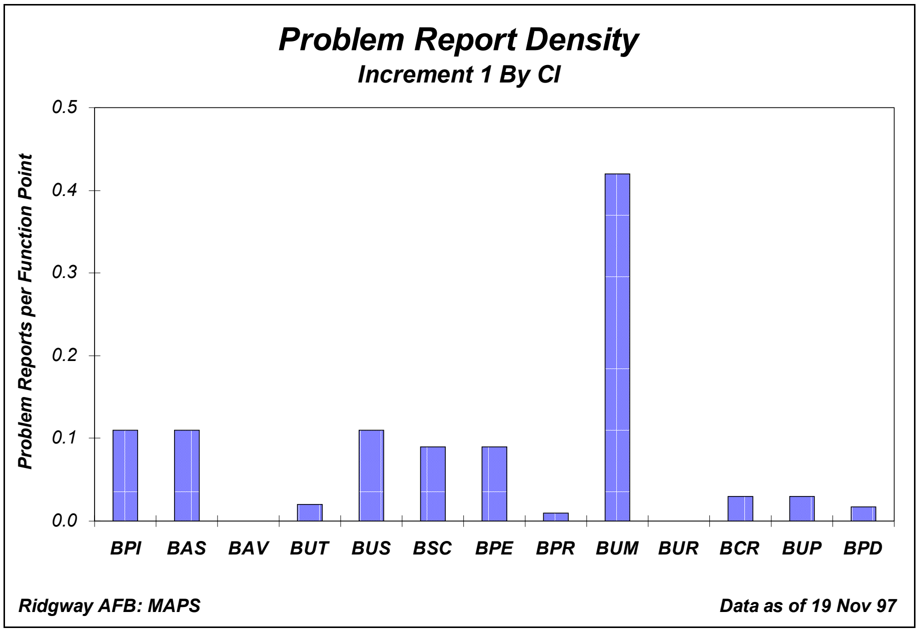


Figure 8B-20. A comparison of problem reports to function points showed that the Unit Mobilization (BUM) CI accounted for most of the defects

Lt. Col. Thompson asked Ms. Cooper to compare how much effort was being applied to correcting the problems with what it would cost to redesign and redevelop the Unit Mobilization CI. Ms. Cooper generated the graph shown in Figure 8B-21 to reflect the effort that was applied over a two-month period.

Ms. Cooper noted that the Unit Mobilization CI required the equivalent of three full-time staff members to support problem resolution. She was surprised that there continued to be such a high rate of newly discovered problems, particularly considering that the Unit Mobilization CI had been in operational use for almost a year. In talking with the lead programmer responsible for maintaining the CI, she found that as existing problems were corrected, new ones were being introduced. She decided to compare the cost of continuing to maintain the CI as currently implemented over a projected ten-year period with the cost of reengineering and maintaining a more reliable version of the CI. The screen and report generation functions did not need to be changed.

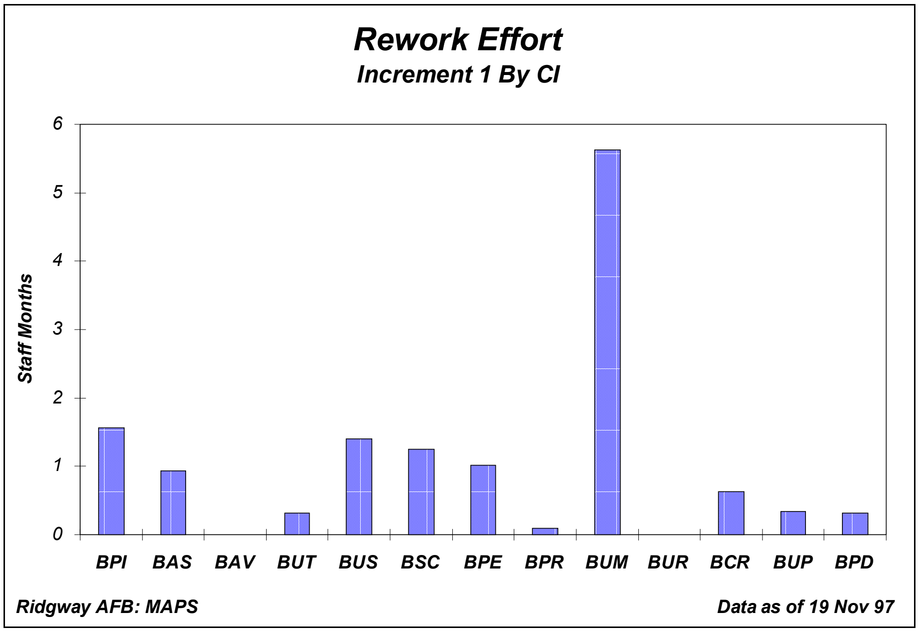


Figure 8B-21. The rework effort indicator also identified the Unit Mobilization (BUM) CI as the major cause of problems

Ms. Cooper estimated that the cost of reengineering would be $1.2 million over a 10-month period, with estimated software support costs of $800K over the remaining nine-year period. This $2.0 million was compared to an estimated $3.0 million cost to maintain the existing CI over the same ten-year time frame. This comparison was based on an average $100K cost per person year.

Lt. Col. Thompson decided to redesign the Unit Mobilization CI and planned to release it in the next MAPS update scheduled for late 2001.

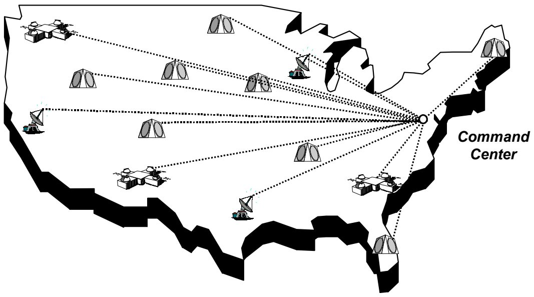
Of course, budget factors needed to be considered in the redesign decision. Given the development funding constraints, it would have been easy to defer the changes. By including the maintenance organization in the decision, funds were made available from several sources to support redesign of the CI during the life-cycle phase.

4.3 Epilogue

The MAPS development turned out to be a good example of implementing a measurement process on an existing project. As the project progressed, the data required to manage the key issues was identified, collected, and analyzed. The measurement activity was focused on the primary software issues of schedule and progress, and product quality.

The measurement process was adapted to the specific characteristics of the MAPS project. Measures suited to information system software, such as function points, were implemented. New measures were also defined to support the installation process. By the end of the MAPS development, the entire project team realized how measurement was useful in identifying and resolving both management and technical problems.

[This page intentionally left blank.]



PART 8C

Sensor Operations and Maintenance Case Study

The Sensor and Operations Maintenance Case Study describes a software support project for a loosely coupled set of radars used by the U.S. Army. This study demonstrates the use of measurement on a project in which the system has been operational for thirty years. Management of this project recently transferred from the operating command to the logistics command. The logistics organization is now responsible for managing all hardware and software configuration changes. The critical issues are largely driven by the age and capacity of the hardware, the current staff's expertise with the technologies, and the harsh environments at the sensor locations. The users require regular software releases to upgrade functionality and to correct problems. Priorities frequently change during the release process. This case study focuses on the use of measurement to improve maintenance cost estimates and to assess and manage the impact of requirements volatility.

The Sensor Maintenance Case Study is organized into three chapters:

- **Chapter 1, Project Overview,** provides an overview of the technical and management aspects of the project.

- **Chapter 2, Tailoring Measures to the Project,** describes the measurement process, data collection problems, and tools used to implement the measurement approach.

- **Chapter 3, Applying Software Measures,** illustrates how measurement helps to objectively estimate the effort associated with each software change request, and the cost of a maintenance software release. Finally, management of requirement changes is discussed.

1

Project Overview

This chapter introduces the sensor maintenance project scenario and describes the technical and management aspects of the project. The project scenario illustrates the implementation of an issue-driven measurement process on an existing maintenance project. Special consideration is given to using measurement data that is readily available within the established software and project management processes. The project is representative of a typical system in operation.

1.1 Introduction

The Space Object Tracking System (SPOTS) is a loosely coupled confederation of ground-based sensors. These sensors provide near-earth orbit space object identification data to command centers throughout the world. Some individual sensors became operational in the 1960's and 1970's. Major computer upgrades occurred as resources were provided; therefore, each site has fourteen sensors with a combination of optical, mechanical, or phased-array technology to provide information to command center operators.

From the time the sensors became operational until 1998, the software was maintained by the command that operated the sensors. The hardware was maintained by a separate logistics organization. According to Ms. Morris, the head of logistics, "This separation between hardware and software caused configuration management problems and made some modifications difficult. For example, in the course of a communications upgrade, both hardware and software needed to be changed. Coordinating between the two organizations and developing an accurate cost and schedule estimate was difficult."

Furthermore, because software maintenance was not the highest priority of the operations command, software releases were not actively planned or managed. Each software change request was developed individually through unit test by a contract engineer and integrated into the operational system without coordination or formal management. This ad-hoc management style led to a number of problems, including:

- Loss of configuration control
- Introduction of a large number of defects into the software
- Changing the same code was changed multiple times during an update
- No clear schedule for software updates
- No clear cost of each release

Upper-level Army management recognized these problems. To address these issues, in 1997 General Knopf and General Texter signed a management directive that called for a reorganization and a new management strategy. The reorganization transferred software management responsibility from the operations command to the logistics organization. The directive also clearly defined the roles and responsibilities for software maintenance between the two groups. Figure 8C-1 shows the roles and responsibilities defined in the directive. Furthermore, to foster a culture of more objective management, the directive stated that measurement would be used to ensure that the goals of the reorganization were met.

Responsibilities for Operations and Maintenance	
User/Operations Tasks	***Logistics Software Team Tasks***
Identify, prioritize, and approve new requirements	*Fix emergency problems*
Validate operational capability	*Identify problem causes, and determine proposed solutions*
Determine if a software release is suitable for use	*Provide user with solution options, cost estimates, recommendations, impacts to other systems, and perceived risks*
Prioritize and approve installation schedule	*Design, develop, and modify software*
Identify problems (e.g., error codes, messages, and incorrect outputs)	*Provide certified release package (e.g., version description, installation procedure, operator checklists, training, etc.)*
Restore system to operations	*Provide necessary test software or data*
Perform configuration management	

Figure 8C-1. Measurement would ensure that the goals of a reorganization and new management strategy would be achieved

With the retirement of Ms. Morris, Mr. Mike Smith was promoted to lead the Project Management Office (PMO) for SPOTS. He was chartered to implement the new management directive. Mr. Smith earned a Bachelor's degree in engineering from the University of Washington and spent several years working on the structural engineering of antennas and radomes, where measurement is common practice. He spent the last eight years working on hardware modifications to the SPOTS sensors. He recently completed the Defense Systems Management College (DSMC) program manager's course.

Mr. Smith organized the SPOTS Project Management Office into eight Integrated Product Teams (IPTs), one for each of the eight sensor types. Each sensor IPT had at least one system engineer, one equipment specialist, one sensor manager and several software engineers. An additional team focused on the development of policies and procedures for configuration management.

1.2 System Architecture and Functionality

The primary objective of the SPOTS maintenance organization is to incorporate user-requested changes to improve the performance, functionality, and/or usability of the system, based on mission requirements. Figure 8C-2 shows the hardware connectivity for SPOTS. The current system consists of eight separate radar types residing at fourteen locations. The radars use large mainframe computers with applications written primarily in Ada, FORTRAN, and various assembly languages to process the target data. Few of the sensors have graphical user interfaces; most rely on text-based messaging and command line systems.

The operating concept of the current system is to receive tasking from the command center, to plan the search for a requested object, and to provide the object's range and location data to the command center. As with many legacy systems, the current implementation has a significant number of problems with computer resources and usability. These problems result in downtime and user errors that delay processing tasks from the command center. The outdated communication system uses antiquated data rates, algorithms, and media to transfer data from the radar site to headquarters. This further impacts the reliability of SPOTS. Consequently, there is a large backlog of change requests for improved functionality, more reliability, and timely processing of tasks.

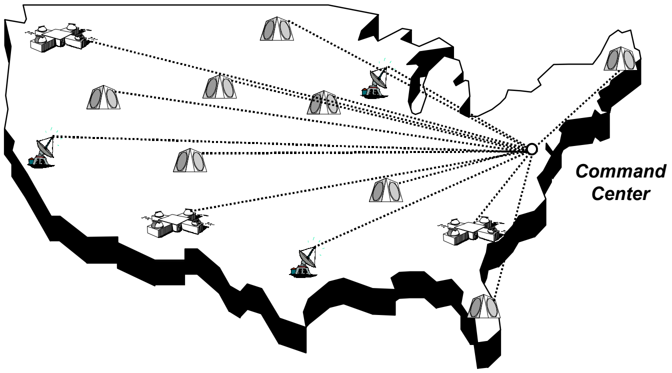


Figure 8C-2. The current system architecture uses antiquated data transfer from the radar sites to headquarters

2

Tailoring Measures to the Project

At the time of the reorganization, software for each of the eight sensors was maintained under a separate contract. Thus, Mr. Smith had to manage eight different contracts for the software. His first action was to consolidate the eight level-of-effort contracts into a single task-order contract to simplify contract management. This approach facilitated negotiation of individual release cost, schedule, and content in accordance with the management directive.

With this action, Mr. Smith immediately faced questions from the user/operations organization, oversight organizations, and headquarters regarding software releases:

- How much will a release cost?
- How long will a release take?
- What changes will be in the release?
- How much rework is occurring?
- How long does it take for a priority change request to become operational?

Additionally, Mr. Smith had a set of questions about the software responsibility he had inherited:

- How much software currently exists?
- How many approved change requests of each priority are awaiting implementation?
- How many new change requests arrive for analysis each quarter? How much analysis effort is required for each one?
- How many emergency, urgent, and routine changes occur?
- How much sensor downtime is caused by software?
- How often does the existing system fail? How many of these failures can be attributed to software?

In response to these questions, Mr. Smith placed Major Alan Richardson in charge of software and hired Ms. Gail Jackson as a measurement analyst. These two individuals were assigned to develop a repeatable software maintenance release process with a measurement strategy that addressed these basic issues.

2.1 Development of the SPOTS Measurement Strategy

As the first step in defining the SPOTS measurement strategy, Major Richardson and Ms. Jackson held a series of meetings with the sensor commanders, headquarters personnel, and SPOTS management to solicit and discuss issues and concerns. After the meetings, Major Richardson and Ms. Jackson organized and prioritized the issues. The prioritization was based on their judgement and the number of people who had raised the issue during their meetings. Next, they defined measures to address the identified issues. The results are shown in Figure 8C-3.

Using Figure 8C-3, Major Richardson and Ms. Jackson identified the data necessary to calculate the identified measures and noted whether the data was currently available. By reading contractor- and site-produced documents and interviewing operations and headquarters personnel, Ms. Jackson identified some historical data from previous software releases. While there was little cost data, there was ample reliability information, some schedule and requirements volatility data, and change request data. Figure 8C-4 clarifies the data that was available and the process or contractual modifications that were necessary to collect the remaining information.

Project-Specific Issue	*Question*	*Measure*
Lack of configuration control	• **How much software is there?** • **How many approved change requests of each priority are awaiting implementation?** • **What kind of changes are made?** • **What changes will be in the release?** • **What is the performance impact of the changes?**	• **Software size** • **Change requests by backlog, priority, kind, CI, and performance impact** • **Computer resource utilization**
A large number of defects exist in the software	• **How often does the existing software fail?** • **How many defects are identified by the operators during system testing?** • **How much sensor downtime is due to software?** • **How many of the objects requested are reported?**	• **Software reliability** • **Downtime due to software** • **Defects found during operational test and evaluation** • **Objects tasked and reported**
Multiple changes to code in consecutive updates	• **How often is a particular CI changed?** • **How many change requests are rejected by the operations approval board?**	• **Changes per CI** • **Change requests rejected**
No clearly defined schedule for software updates	• **How long does a release take?** • **How long does a priority change request take to become operational?** • **How much do the requirements change during a release?** • **What percent of schedules are met?**	• **Schedule including:** - **release plan approval to release acceptance** - **change request written to approval to delivery** • **Requirements added, deleted, and changed after release approval**
No understanding of the release cost	• **How much does a release cost?** • **How much effort does it take to analyze a change request?** • **What change requests require the most effort to implement?** • **How many change requests are evaluated each month?**	• **Cost per release** • **Effort to:** - **Analyze the impact of a change request** - **Implement per change request type** • **Change requests evaluated**

Figure 8C-3. A listing of SPOTS Issues/Questions/Measures was used to define useful measures

Measure	*Data Needed*	*Data Available*
Software size	**Source lines of code by sensor, platform, and language**	**No. A special task order will be issued to develop software inventory.**
Change request by backlog, priority, kind, type, performance impact, and effort	**Change requests approved by:** • **Priority - emergency, urgent, or routine** • **Kind - modification or fix** • **Type - define "type" including rules** • **Performance Impact**	**Some. All change requests are available, including the priority and kind. Meeting minutes from the operations approval board contain the date and disposition of requests. The types need to be defined. Individual task orders need to be modified to get the remaining data.**
Computer Resource Utilization	**CPU busy, memory, disk space, and I/O throughput pre- and post-release**	**No. Individual task orders will be modified to collect data important to each sensor.**
Software reliability and downtime due to software	**Failures caused by software, and system operating hours**	**Yes. Downtime incidents are logged, and all problem reports are classified as a failure or enhancement. Operating hours are logged in a system scheduling document.**
Defects found during operational test and evaluation	**Problem reports written during user scenario testing**	**Yes. Operators complete problem reports during testing and produce a letter after the test.**
Objects tasked and objects reported	**Objects requested by Control Center, and objects reported by sensor**	**Yes. The user organization reports this data on a daily basis.**
Changes per CI	**Name of each affected configuration item, and number of changes made to that CI**	**No. A software configuration management system will be implemented by the CM team to automatically gather this information by release.**
Change requests rejected	**Number of change requests reviewed and withdrawn by the board**	**Yes. Meeting minutes from the board are published after each meeting with the disposition of each change request reviewed.**

Figure 8C-4. The selected SPOTS measures were reviewed to determine if the data was currently available

Measure	Data Needed	Data Available
Schedule including: - Release plan approval to release acceptance - Change request written to approval to delivery	• Date the release is approved, delivered, and approved • Date the change request is written, approved, and delivered	Some. Some sensors provided planned release delivery dates. Historical release start dates may be derived from reading documents. Individual task orders will be modified to collect this data. Change request dates are available.
Requirements added, deleted, and changed after release approval	Planned number of change requests, and updates to plan when a change occurred	Some. Some sensors provided planned and actual delivery content. Some changes could be derived from reading documents. Individual task orders will be modified to collect data.
Cost per release	Dollars spent (by activity) for all personnel involved on each release	No. Individual task orders will be modified to collect data from the implementing contractors, and the government team will be asked to complete data collection forms.
Effort to: - Analyze the impact of a change request - Implement the change request by type	Engineering hours spent per change request by type of change	Some. The government engineer responsible for performing the impact analysis will be required to record the effort spent on each impact report. The categorization by type will be developed, and individual task orders will be modified to collect the effort data from the implementing contractors.
Change requests evaluated	Number of change requests reviewed by the board	Yes. Meeting minutes from the board are published after each meeting, with the disposition of each change request reviewed.

Figure 8C-4 (continued). The selected SPOTS measures were reviewed to determine if the data was currently available

Mr. Smith and the sensor IPTs were briefed on the analyses results summarized in Figures 8C-3 and 8C-4. Ms. Jackson was able to answer many of the questions raised during the meetings by compiling the available data into a set of spreadsheets. This quick demonstration of measurement value led to the formalization of the measurement effort. Ms. Jackson was asked to develop requirements for data collection on all software releases. Ms. Jackson's response was a memo calling for the following data to be included in all software release task orders:

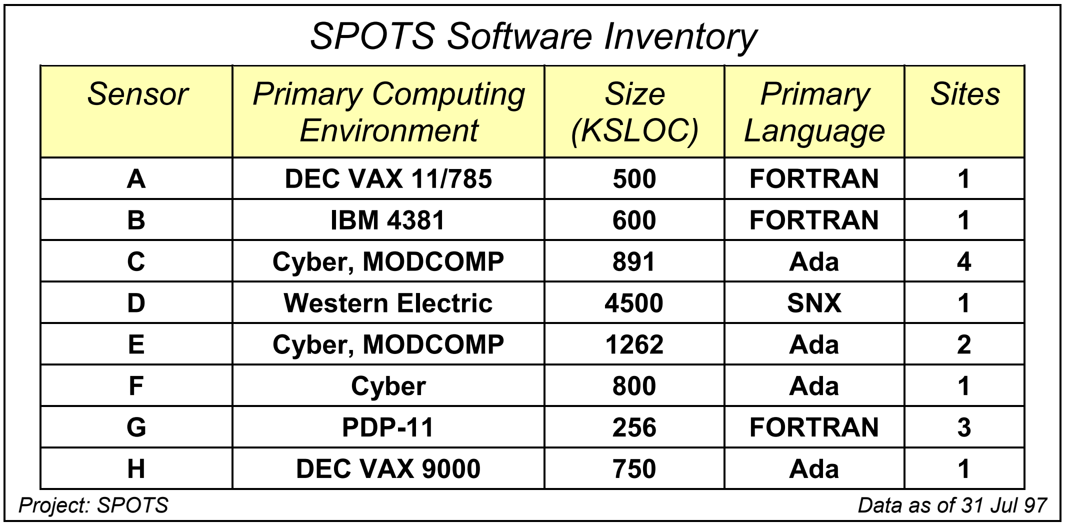
- Cost Data
  - Effort expended per change request and per month
  - Categorization of change request according to the types that were defined
  - Dollars expended per month
- Schedule Data
  - Milestone dates for each change request and for the release as a whole
  - Milestone dates for each change request, including requirement analyzed, design complete, code and unit test complete, and integration complete
  - Milestone dates for the release, including release date, start date and release installation date
  - Plan and actual dates for each milestone
- Growth and Stability Data
  - Planned number of change requests for the release, and all changes made to the release content (change requests added, deleted, or modified) during the release
  - Actual logical source lines of code added, changed, or deleted per change request and per release
- Computer Resource Utilization
  - Pre- and post-release data relative to a specific sensor (some sensors are limited by memory, while others may exceed the required response time)

The government personnel were also required to collect data on their workload, including effort spent analyzing software changes and time spent on a particular release.

2.2 Implementation of SPOTS Measurement Strategy

Major Richardson and Ms. Jackson published a measurement guidebook for use as a reference by the SPOTS team. The guidebook explained how the data was collected based on the software maintenance release process, as well as which measures would be used together for integrated analysis. Major Richardson and Ms. Jackson also developed a one-hour briefing to communicate the goals, use, and expectations of the measurement project to the sensor IPTs and the implementing contractors. They began to integrate a set of tools to support measurement collection and analysis including tools for data storage, change request tracking, configuration management, cost estimation, and software reliability modeling.

To fully implement the strategy, the missing data needed to be collected. Some of the unavailable data was relatively easy to generate. For example, a contract was awarded to baseline the software inventory. The contractor reported that the system contained a total of eight million source lines of application code written in more than 20 different languages to execute its mission. Figure 8C-5 summarizes the SPOTS software inventory.



SPOTS Software Inventory

Sensor	Primary Computing Environment	Size (KSLOC)	Primary Language	Sites
A	DEC VAX 11/785	500	FORTRAN	1
B	IBM 4381	600	FORTRAN	1
C	Cyber, MODCOMP	891	Ada	4
D	Western Electric	4500	SNX	1
E	Cyber, MODCOMP	1262	Ada	2
F	Cyber	800	Ada	1
G	PDP-11	256	FORTRAN	3
H	DEC VAX 9000	750	Ada	1

Project: SPOTS

Data as of 31 Jul 97

Figure 8C-5. SPOTS contained eight million source lines of application code in more than 20 different languages

In some cases, the necessary data was available. For example, the sensor IPTs were interested in tracking the software failure rate in the field to understand how many problems were affecting the sensor operators, and how well the delivered releases were performing. Failures were counted whenever the system performance did not meet user requirements, such as missing a large number of tasked observations, or when the system crashed. A sensor operator inspected each problem to determine if a software failure had occurred. All downtime incidents caused by a software failure were counted. These data items and explanations were reported in monthly maintenance logs provided by the site operators. Major Richardson needed to identify which software version was operating at the time of the incident. Figure 8C-6 shows the failure rate for two releases prior to the reorganization and seven releases after the directive was implemented. Over the nine releases the number of operational failures was reduced from 6.8 per 1,000 operational hours to fewer than 2.4 per 1,000 hours. This improvement in product quality is also visible in the work backlog chart of Figure 8C-7. Users generated fewer change requests when the system operated without failure for long periods.

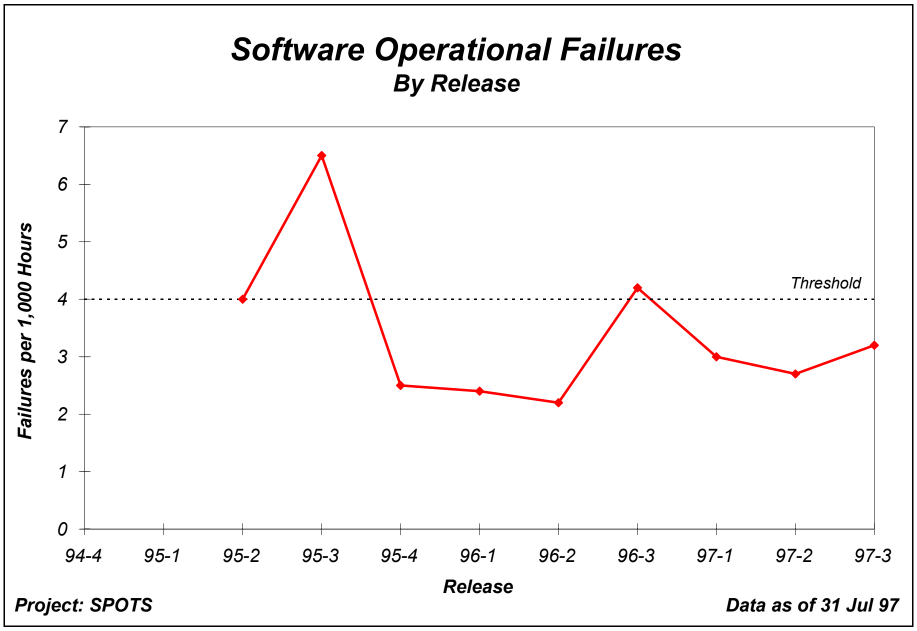


Figure 8C-6. In the last nine SPOTS software releases, the number of operational failures was decreased by more than half

The sensor IPTs used the failure rate information in several ways. One was to set a threshold at four failures per 1,000 hours of operation, based on the historical data prior to the reorganization. If the system was operating below this threshold, a sensor IPT manager could decide to incorporate more difficult changes into the next release. If the system was operating above the threshold, the decision could be to switch to a previous version of the software or to only allow fault corrections until the failure rate was again below the threshold.

Second, the graph was used to estimate the expected number of failure events during a mission and the probability of completing a mission of a certain duration without a failure. For example, assume that the control center was planning a one-week observation period and needs to know the probability of the software delivery supporting operations for the entire period. In this case, the probability of a failure-free mission is given by the standard formula:

The reliability (R), or probability of no failure, in time period t:

$$\mathbf{R = exponential\ (-\lambda t) = e^{(-\lambda t)}}$$

Where λ is the failure rate from Figure 6c.2-4 (e.g., 2 failures/1000 hours) and t is the 168 hours in a week. Therefore, the probability of no failure in one week is:

$$\mathbf{R = e^{(-0.002*168)} = 0.71}$$

The reliability value of 0.71 means that there is a 71-percent chance that the system will not fail in one week of operation because of a software problem.

In some cases, the available data required much time and effort to validate, understand, and use. For example, to find out how many change requests were currently open, Major Richardson read the minutes of the operations approval board, reviewed historical release documentation, and contacted each sensor site. Several discrepancies between the sets of records were uncovered, such as change requests that the board showed as not evaluated, the site showed as approved and not complete, and the records showed as delivered. After spending several weeks cleaning up the multiple databases and coordinating with the operations personnel, the corrected data was entered into a series of spreadsheets.

Figure 8C-7 was developed to show the total inventory of change requests by priority that existed on SPOTS. Each decrease in the work backlog (Feb 00, Sep 00) corresponded to an accepted delivery. None of the sensors had outstanding emergency change requests. This change request indicator was also decomposed by sensor for more detailed analysis. Mr. Smith requested that this indicator be updated monthly to keep him apprised of the change request workload and status. Detailed analysis and external reporting were done quarterly. This allowed enough time to show trends in the data over the typical six-month maintenance cycle. Detailed analysis on a monthly basis would not allow the smoothing to occur naturally and might cause unnecessary alarm within the operations community. To support this regular reporting, a central measurement database was developed to maintain the status of all approved change requests.

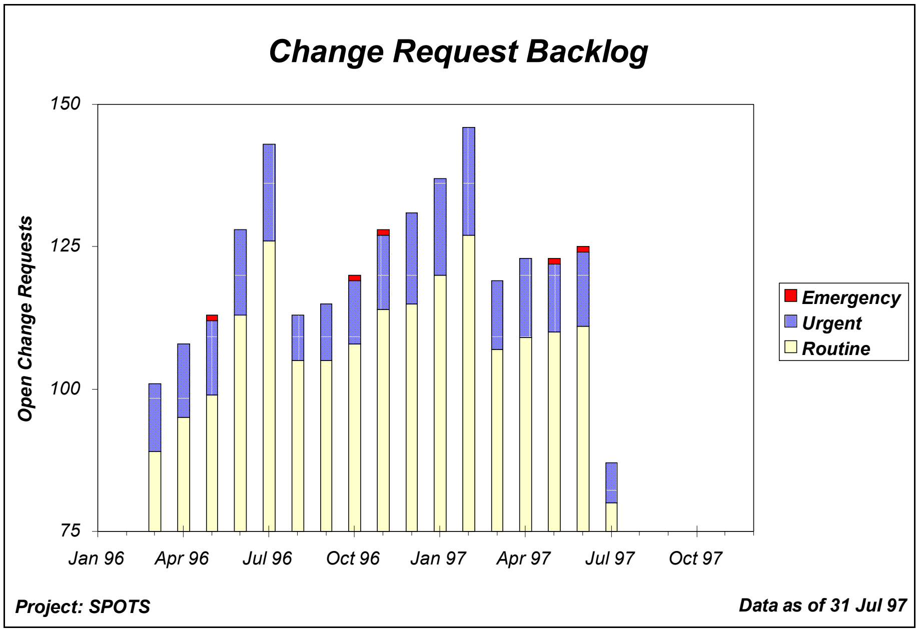


Figure 8C-7. The change request backlog indicator reported the total inventory of change requests by priority

At other times, the analysis requirements dictated using both existing data sources and creating new collection forms. For example, Mr. Smith needed to understand the additional workload on his engineering staff because of the reorganization. Ms. Jackson was asked to address this issue. She decided to examine the amount of yearly effort required to evaluate and estimate costs for incoming change requests. She needed to know three things: 1) the rate at which change requests arrived, 2) the time spent evaluating the change, and 3) the number of changes that were rejected by the user organization after evaluation. At any time during the software change process, the user board can withdraw an approved change request. Another change may have corrected the problem or an external event, such as a site closing or an external interface change, may have invalidated the problem. The number of change requests that were withdrawn out of the total evaluated was a measure of rework in the process. Using the minutes of the users' requirements control board, Ms. Jackson collected data on incoming change requests and rejections for several quarters. The results are shown in Figure 8C-8.

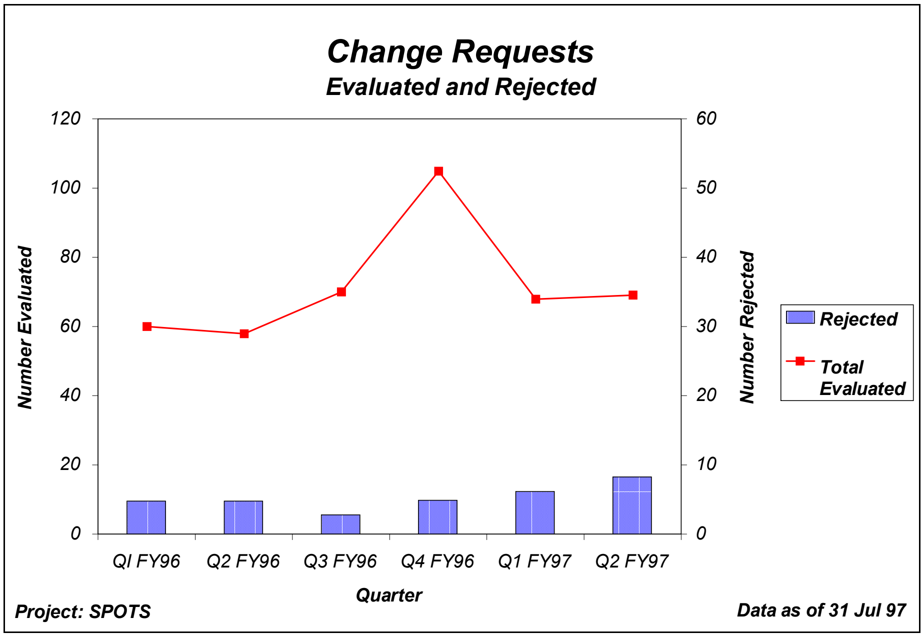


Figure 8C-8. The incoming change request indicator showed the workload on the government engineering staff

Ms. Jackson also asked the engineers to record the number of hours they spent on each evaluation. The average, based on one year's worth of evaluations, was 5.5 hours per change request. From Figure 8C-8, 72 change requests were evaluated each quarter, an average (288 per year). Thus, the engineering staff expended an additional 1,584 hours of work (about 0.75 staff) per year on evaluations. Of these, an average of 8 percent (23 evaluations) were withdrawn each year. These withdrawn changes translated to a loss of only $7,590 per year (5.5 hours per evaluation at $60 per hour for 23 withdrawn rework evaluations per year). Rework is not a priority issue for the SPOTS change analysis process.

2.3 SPOTS Measurement Tools

To ensure that the measurement data was collected consistently across all releases, and to ensure that it could easily support management decisions, Major Richardson and Ms. Jackson identified a standard suite of tools for use by analysts and project personnel. These tools support the analysis of the measurement data to answer the questions and address the issues. Although the tools are neither completely automated nor integrated, Ms. Jackson, Major Richardson, and the sensor IPTs have found them useful for SPOTS.

Recognizing that integration of several measures was required, the tools needed to collect, store, and manage data from multiple sources. In the SPOTS environment, the tools required a data repository element, a cost/resource estimation tool, a configuration management tool, a software change request tracking tool, and a reliability estimation tool.

The data repository is a commercially available spreadsheet running on a desktop computer. The spreadsheet allows the IPTs to view historical data from the software maintenance releases as well as track planned and actual measures for ongoing releases. It provides graphical displays of the measurement data, as well as linear regression or tabular analysis of the data.

Cost and schedule estimates are assisted by an implementation of Boehm's Constructive Cost Model (COCOMO) specifically tailored for software maintenance. This tool helps a release manager estimate the effort, cost, and schedule required to produce a software maintenance release. Major Richardson and Ms. Jackson used historical data to calibrate the model to their environment.

Another tool used is the configuration management system. This tool allows the analyst to identify the configuration item changed, as well as the date and the size of an individual change.

The Change Request tracking system is a PC-based database tool that allows each IPT to track the status of all change requests.

Finally, the Computer Aided Software Reliability Estimation (CASRE) tool is available for software reliability assessments. (This tool is a public-domain project available in the McGraw-Hill *Handbook of Software Reliability*, Michael Lyu, editor.) This tool allows IPT engineers to forecast field failure rates from test data and track operational failure rates.

3

Applying Software Measures

With the reorganization, the SPOTS sensor IPTs now had the responsibility to estimate the impact and cost of individual change requests. They were also responsible for estimating and tracking the cost of individual software releases. Because these were high priority issues and little historical data was available, Ms. Jackson and Major Richardson decided additional measures were required.

3.1 Estimating Effort on an Individual Change Request

Originally, the effort for an individual change request was estimated by the analyst or engineer assigned to the evaluation. The analyst or engineer performed a design analysis and estimated how long it would take to code and integrate the design. This produced inconsistent and variable estimates for each change, estimates that were usually ignored in the release planning process.

To improve this procedure, Ms. Jackson and Major Richardson categorized 178 completed software changes into ten types:

- Computational
- Data Handling
- Improvement
- Input
- Interface
- Logic
- Operations
- Output
- Performance
- Specification

Their plan was to find the effort associated with each class of change and base future estimates on the historical data. The sample consisted of 67 enhancement-related changes (38 percent) and 110 defect corrections (62 percent). Using the categories, Major Richardson prepared a bar graph of this change data, as shown in Figure 8C-9. Based on the data, the majority (65 percent) of the implemented changes were found to be logic, data handling, system improvement, and system mission enhancement. No changes out of the 178 impacted the SPOTS input type. This data led to changes in the inspection process to concentrate on logic and data handling problems. The next questions concerned where the engineering effort was being expended, and if it correlated to the types of problems found.

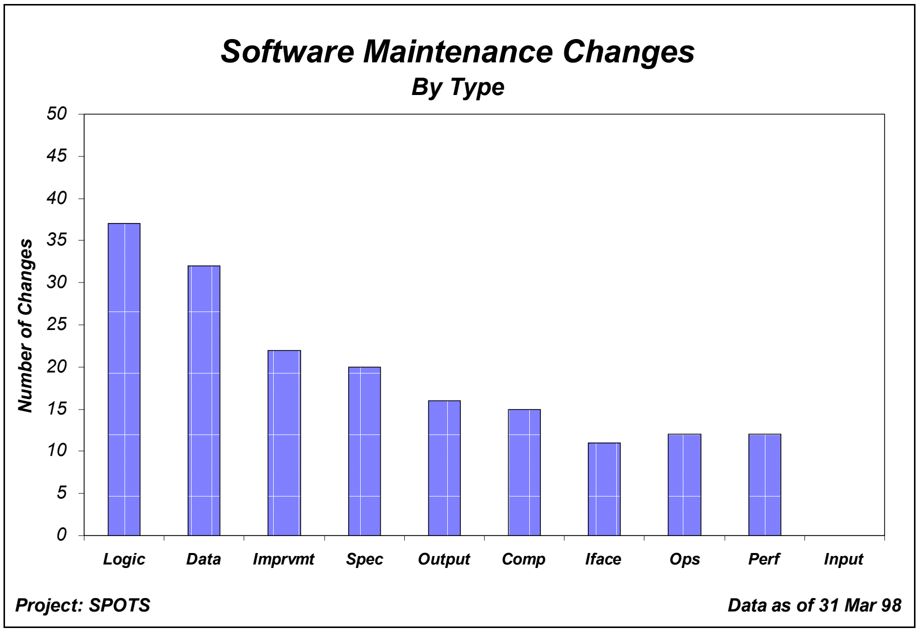


Figure 8C-9. Plotting maintenance changes versus functions caused the inspection process to concentrate on logic and data handling problems

Because the release for each change request was managed as a separate project, Major Richardson was able to collect data from the contractor on the engineering effort associated with each change. Using the change types and aggregating the effort related to each change, he created Figure 8C-10. This figure shows that, although changes based on requirements or interface specification changes ranked fourth in number of changes with 22, they accounted for 42 percent of the total effort at 582 staff-days. Logic changes required the third largest amount of effort. This historical data is integral to the cost estimating procedure developed by Ms. Jackson to support release planning and change request impact analysis.

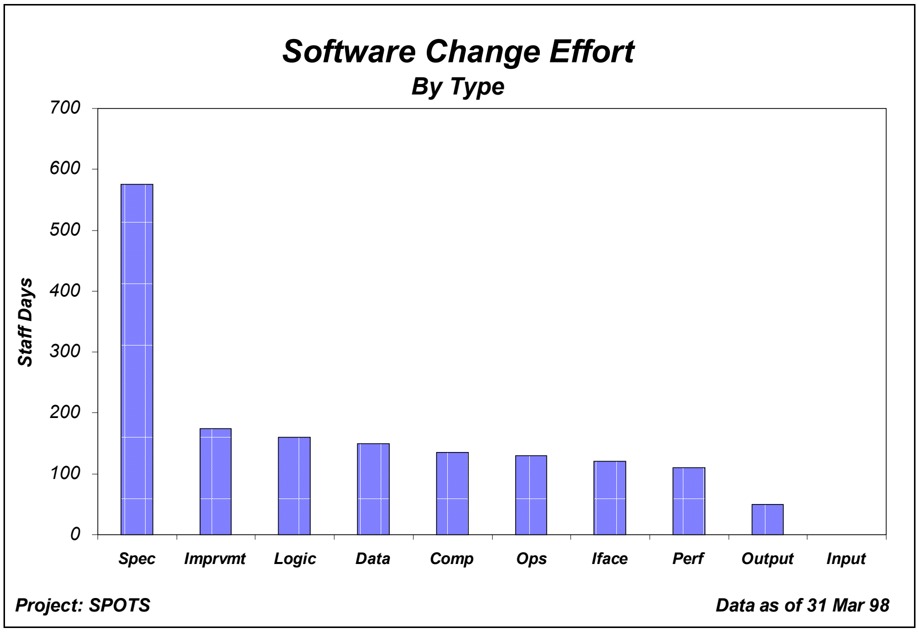


Figure 8C-10. An indicator that aggregates staff effort to the types of change caused a change in cost estimating for future releases

The data from Figures 8C-9 and 8C-10 can also be combined to calculate the average staff-days of effort required to implement each type of change request. By reviewing the change requests and accurately assigning them to the change types established by Ms. Jackson and Major Richardson, sensor managers were able to estimate the engineering staff-days required to design, code, and test individual changes. For example, interface specification changes required an average of 36 staff-days of effort, with a range between 20 and 100 staff-days. The average staff-days of effort needed for requirement specification changes was 22, with a range between 3 and 75 staff-days.

As each release was completed, each change was categorized and the actual effort for each was added to the database to refine the estimates for the next set of changes.

3.2 Estimating Cost on a Software Maintenance Release

The sensor IPTs work with the sensor operators and Control Center users to define those changes that will be included in each release. As a part of this analysis, an expected cost and schedule for the release is estimated. If the expected cost, schedule, reliability, and functionality are acceptable to all parties, a task order is sent to the implementing contractor for a bid proposal.

The contractor usually provides a proposal to implement each release, including an organization chart and functional work allocation (e.g., for change request Y, a senior engineer will work 50 hours and quality assurance will spend 100 hours ensuring completeness of deliverables). The team is generally small, comprised of no more than seven or eight people. The contractor's proposal is a bottom-line effort, and cost is usually all that is reviewed.

Toward the end of the first year of the measurement process, a release was defined and sent out for a bid from the contractor. The bid was received at 16,000 staff-hours for the release. This seemed high to the sensor teams, but they did not have a good technique for evaluating the bid. Because of the measurement strategy, Ms. Jackson had data on four previous software releases for this sensor. Using this data, she developed the cost estimation procedure shown in Figure 8C-11 for comparison with the bid. Using this procedure, three cost estimates were derived: 7,500 hours, 8,000 hours, and 6,250 hours. After discussing and clarifying the original 16,000-hour estimate, the cost of the release was negotiated to be 8,500 staff-hours, a $500,000 savings over the original bid. The actual release cost included 8,625 staff-hours.

Cost Estimation Procedure

Introduction

The Software Maintenance Release Cost Estimation Procedure involves making three estimates and using engineering judgment to arrive at a final forecast. Two of the three estimates are based on empirical data collected from several software releases. The third estimate is based on a parametric cost estimation model called Constructive Cost Model (COCOMO). The COCOMO model has been calibrated based on previous maintenance releases of this project.

For the purposes of this procedure a software maintenance release contains more than one change. The following paragraphs describe each step in the procedure.

Estimate #1: Estimate engineering effort for individual changes

1) Classify the proposed Change Request according to type.
2) Record the average and maximum effort required for each change request based on historical data.
3) Sum the average and maximum effort estimates for the entire release and multiply by 2 to account for the non-engineering overhead effort.
4) Multiply the staff-effort by $110/hour to estimate the cost of the release.

Estimate #2: Estimate release cost based on previous releases

1) Examine the historical release database and compare the site, size (number of changes), planned schedule, and complexity estimate of this release with those in the database.
2) Record the average and maximum cost for the “best match” releases.

Estimate #3: Run the COCOMO Model to estimate release costs

1) Create a new maintenance project in COCOMO.
2) Enter the required common parameters such as size of each configuration item being modified in this release, development mode, and labor costs.
3) Enter the required Cost Driver Data such as complexity and staff experience.
4) Enter a best guess at the schedule data.
5) Run the COCOMO reports and use the resulting cost estimates.
6) The COCOMO based tool has a “what-if” function to examine constraints on staff-size and schedule to recalculate costs under special circumstances. Use this feature to analyze the release assumptions.

After all three cost estimates are calculated, examine the range of costs. Think about the contents of the configuration control briefing such as reliability, maintainability, backlog, and complexity, and make judgments about how these factors will influence the cost of the release. Estimate the release based on the available information. This approach helps to justify the estimates and helps save money in negotiations with the contractors.

Figure 8C-11. Measurement data on previous software releases allowed the acquirer to develop an accurate cost estimation procedure

3.3 Measuring Requirements Volatility

Another key issue that drove the reorganization was the lack of clearly defined schedules for software releases. Because each SPOTS release is managed as a separate task order, the estimated schedule is provided with each proposal. The early schedules were not met. To investigate this issue, Ms. Jackson reviewed data from several releases and interviewed IPT managers, operators, and contractors. She determined that after the sensor IPT managers, operations team, and Control Center team agreed on a delivery plan, changes to the requirements became a major issue. Mr. Smith asked Ms. Jackson to look into requirements volatility and to devise a way to estimate the impact of requirements changes on release schedules.

Ms. Jackson defined requirements volatility as change request additions to the delivery content, change request deletions from the delivery content, or changes in the scope of an existing change request. A change in scope meant the estimated effort increased by more than 25% from the agreed-upon plan. The requirements volatility for 21 deliveries is shown in Figure 8C-12. Only five of the ten most recent deliveries experienced unstable requirements, while ten of the first eleven releases experienced a problem. This reduction is partially due to management’s focus on this issue. Release 17 experienced a large requirements change because of a change in the mission for the sensor. The original plan contained five change requests. During design, those five change requests were deleted from the release, and twenty high-priority changes were added. This was a joint decision between the operations team and the development team. Measurement supported this decision by predicting the cost and schedule impact, and showing that there would be no performance impact. The following paragraphs describe how the schedule impact was forecast.

Figure 8C-13 is a scatter plot of schedule performance vs. the percentage of requirements volatility for these deliveries. A 100 percent value means the schedule was met, less than 100 percent indicates early delivery, and greater than 100 percent indicates late delivery. By drawing a trend line, Ms. Jackson was able to predict the size of the schedule slip using the measured volatility of the requirements. The schedule increased regardless of whether the requirement's change is an addition or deletion, because the x-axis is a percentage of all changed requirements.

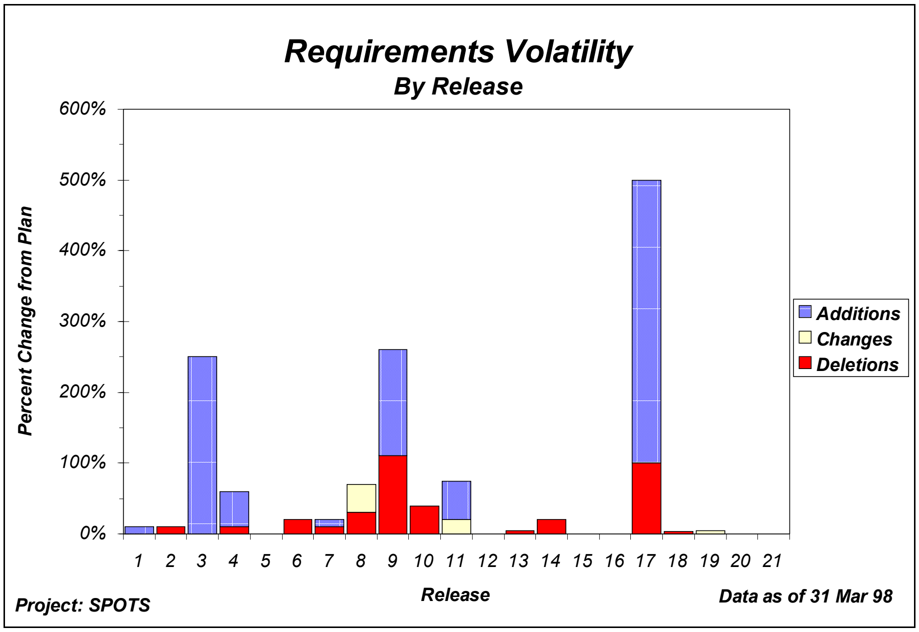


Figure 8C-12. The requirements volatility indicator shows a steady decrease in unstable requirements, due to management's focus on this issue

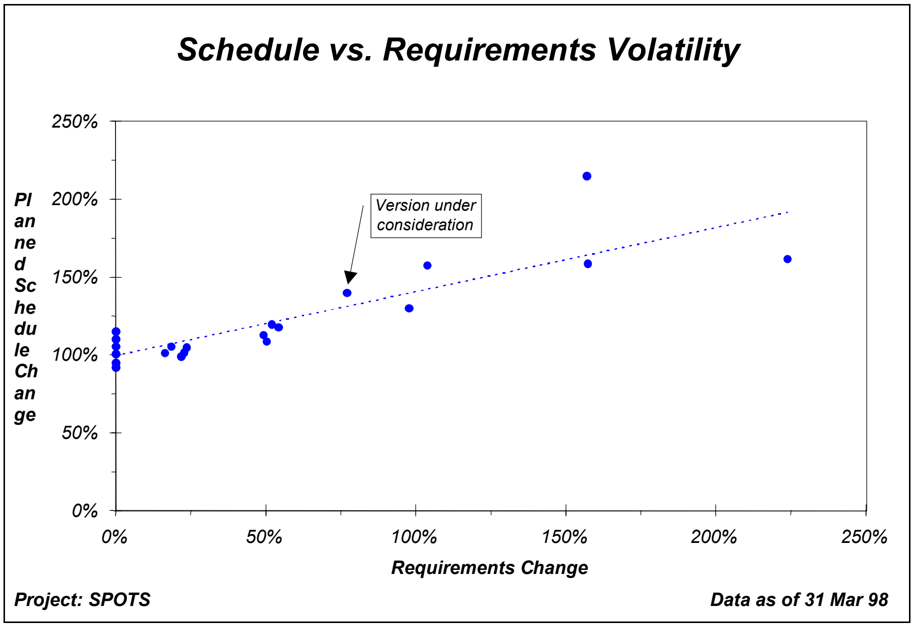


Figure 8C-13. An indicator for schedule performance versus requirements volatility was able to predict the size of an expected schedule slip

The graph in Figure 8C-13 helped Ms. Jackson explain the expected impact of changes to the delivery plan as they arose. For example, one version release that was scheduled for delivery in 91 calendar days contained 15 planned change requests. The operations group wanted to add five changes requests, drop two of the existing ones, and change the scope of four at preliminary design. Using the graph, Ms. Jackson forecasted the impact to be a 30 percent schedule slip, or approximately 27 days added to the original 91-day schedule.

Upon seeing the graph and prediction, the operations group decided that this slip was not acceptable and that only the scope changes would be incorporated. The graph and its interpretation facilitated objective communication about version release plans and benefited customer relations.

3.4 Epilogue

The measurement process has provided substantial benefit in addressing the original issues identified by the Army management. The selected software measures have clearly addressed the majority of the project issues, and have provided information to support project management decisions.

The sensor software configurations are under control. The IPT managers and Mr. Smith are able to report the current operating versions, the size of the inventory, the Configuration Items (CIs) that are changed, and the frequency of changes. The measurement activity played a large part in this.

The failure rate of the software in the field has diminished since the reorganization, and the change request backlog has also been reduced. Furthermore, the teams are capable of forecasting the failure rate prior to a release, allowing better planning.

All releases now have clearly defined schedules that can be monitored and assessed for feasibility. Also, the impact of changes to the release content can be evaluated and discussed objectively with all parties involved in SPOTS.

Most importantly, the sensor managers, Mr. Smith, and Army managers have a better understanding of the release costs. The cost of individual change requests can be estimated. The number of change requests submitted and the effort required to analyze a change request is known, allowing better management of the SPOTS workload. Finally, the amount of rework can be measured on both the change request approval activity and the release implementation process.