Practical Software Measurement, Performance-Based Earned Value

Paul Solomon
Northrop Grumman Corporation

Successful software project management can be achieved by focusing on requirements, selecting the most effective software metrics, and using Earned Value Management. Best practices and lessons learned by the Northrop Grumman team in developing weapons system software for the B-2 Stealth Bomber are discussed.

This article discusses a set of integrated, performance measurement techniques that increased Northrop Grumman Corporation's software success. These techniques can enable excellent project management in the following ways:

- Defining effective metrics for sizing the project and measuring progress.
- Using Earned Value Management (EVM) as the key, integrating tool for control.
- Defining quality goals in terms of project milestones and metrics.
- Planning for incremental releases and rework.
- Revising the plan for deferred functionality and requirements volatility.
- Focusing on requirements, not defects, during rework.
- Using testable requirements as an overarching progress indicator.

These techniques are based on the following industry and professional standards:

- CMU/SEI-92-TR-19, Software Measurement of Department of Defense Systems.
- "A Guide to the Project Management Body of Knowledge (PMBOK)," Project Management Institute, December 2000.
- Practical Software and Systems Measurement: A Foundation for Objective Project Management (PSM) [1].
- (ANSI/EIA)-748-98, EVM Systems Standard (Standard), American National Standards Institute/ Electronics Industry Association.

The techniques evolved from lessons learned and continuous process improvement during development of embedded weapons system software for the U.S. Air Force B-2 Stealth Bomber and other programs at Northrop Grumman Corporation's Air Combat Systems (ACS), a business area of the company's Integrated Systems Sector. The ACS software organization achieved Level 4 using the Software Engineering Institute's (SEI) Capability Maturity Model® (CMM) in 1998 and has a goal of achieving Level 5 in 2001.

The essence of EVM, per the Standard, is that at some level of detail appropriate for the degree of technical, schedule, and cost risk or uncertainty associated with the program, a target value (i.e., budget) is established for each scheduled element of work. As these elements of work are completed, their target values are earned. As such, work progress is quantified and the earned value becomes a metric against which to measure both what was spent to perform the work and what was scheduled to have been accomplished. The combination of advance planning, baseline maintenance, and earned value analysis yields earlier and better visibility into program performance than is provided by non-integrated methods of planning and control.

> "The essence of EVM ... is that at some level of detail appropriate for the degree of technical, schedule, and cost risk or uncertainty associated with the program, a target value ... is established."

Improvements in Opportunities

Despite being at SEI CMM Level 3 since 1995 and having a validated EVM system, the software development organization was not consistently achieving its objectives and customer expectations. More importantly, the management control system was failing to accurately report project performance. These issues were identified and disclosed in the quality audits of the EVM organization. In 1996, an audit defined the following issues and goals:

> "The process for managing software projects with regard to baseline planning, determination of schedule milestones, and earned value could be improved to provide better milestones and metrics for interim performance measurement during development, testing, and rework … actual progress against the total technical requirements is not displayed on a schedule in relation to a plan, a projected or actual software release on a schedule may not reflect completion of all effort originally planned, and earned value does not necessarily represent the percentage of completion of the total statement of work.
>
> It is recommended that the Software Engineering Process Group (SEPG) be empowered to develop a better process for measuring and reporting progress on software projects. It is recommended that the following topics be addressed:
>
> - Criteria for determining which planned requirements are significant for tracking progress.
> - Criteria for milestone definitions.
> - Earned value and internal replanning for deferred requirements or functionality.
> - Earned value and internal replanning for revised requirements.
> - Planning and measuring progress during rework phase."

Existing Measure Shortcomings

A team was formed to review existing measures and processes. It found that, although they adhered to company policies and relevant quality standards, including the SEI core set of software measures (size, effort, schedule, and qual-

ity), the measures used were not effective for technical progress.

The first finding was that during the initial coding phase, the most common sizing measure was source lines of code (SLOC). SLOC was utilized as a sizing measure as the basis for budgets and for earned value using a percent of completion method. However, the analysis concluded that there is usually a significant error in estimating SLOC. Consequently, any progress metric based on SLOC, including EV, was highly volatile. For example, all projects reviewed had software components that experienced multiple, significant increases in estimated SLOC. When the new estimate was first used as a denominator for percent complete, then negative progress and earned value were reported.

Second, while the schedule metrics and procedure discussed completion milestones, the milestone definitions and completion criteria lacked quantifiable objectives. Normally, an incremental build is released despite not incorporating all the planned functional requirements. It had been practice to display a completed milestone on the schedule and to take all of the earned value that was budgeted for that milestone without disclosing that not all the base-lined requirements or functionality had been achieved.

Third, the process review disclosed a deficiency regarding product quality measures such as defects found and closed. A manager normally uses a burndown curve of defects or trouble reports and tends to focus on eliminating defects rather than attaining requirements. Earned value had been based on the burndown plan. However, because the presence of defects indicates the failure to meet requirements, measures of defects are not the best measures of progress. Also, any measure of progress based on defects is unstable because the number of defects discovered during reviews and testing is always different than planned, and removal of one defect often results in detection of new ones. There were no metrics to track progress toward meeting all system requirements. As a result, progress measured as the ratio of defects removed to total estimated defects was a more volatile measure than the percent of SLOC completed. Also, there was no budget to enable earned value for the remaining work.

Fourth, the schedule and performance measurement baseline had been predicated on a discrete number of software builds but completion of the project often required many additional builds. However, earned value was taken as originally budgeted when builds were completed. As a result, there was no remaining budget for the additional builds and the cost performance reports overstated schedule status.

Practical Software and Systems Measurement

A good source of metrics is the Practical Software and Systems Measurement (PSM) guide. PSM provides project and technical managers with the quantitative information required to make informed decisions that impact project cost, schedule, and technical performance objectives. PSM is applicable to the overall planning, requirement analysis, design, implementation, and integration of systems and software activities. It provides a process to collect and analyze data at a level of detail sufficient to identify and isolate problems. This data includes estimates, plans, changes to plans, and counts of actual activities, products, and expenditures. The unit level (as defined by the product component structure or system architecture) is the most commonly used level of detail.

Resultant Process Improvements

The set of process improvements had five components:

- Developing the Performance Measurement Baseline (PMB).
- Requirements decomposition and traceability.
- Planning for defects and rework.
- Selection and use of software metrics.
- Performance-Based Earned Value.
- Revisions to plan for deferred functionality.

Performance Measurement Baseline

EVM begins with defining the project's product and management objectives. These technical, schedule, and cost objectives are transformed into a PMB schedule and budget baseline (also commonly called a Work Breakdown Structure). The team developed standard templates for the PMB. The templates ensured knowledge transfer and inclusion of common project components such as key schedule constraints, subcontractor control milestones, and systems engineering activities.

Requirements

During the requirements phase, high-level requirements are defined and decomposed to the levels needed to govern the design, implementation and integration, and test phases. Establishing a time-phased requirements baseline against which progress can be consistently measured is the most important EVM step. It drives the project sizing, the resource forecast (budget), and the schedule. The technical requirements also establish the criteria for completing tasks. The output of the requirements phase defines the criteria or attributes for completing significant milestones, or taking earned value in all subsequent levels and stages of development. Of equal importance are a disciplined requirements traceability process and a requirements traceability data base.

To ensure the acceptance of the end product and enable consistent performance measurement, allocated requirements should be testable and traced to detailed specifications, software components, and test specifications. Per PSM, some requirements may not be testable until late in the testing process, others are not directly testable, and some may be verified by inspection.

Dr. Peter Wilson, of Mosaic, Inc., discusses the utility of testable requirements [2]. Per Dr. Wilson, a testable requirement is one that is precisely and unambiguously defined, and one for which someone must be able to write a test case that would validate whether or not the requirement has or has not been implemented correctly. The number of testable requirements may be very different from the number of test cases. There are a number of reasons for this:

- A testable requirement may require more than one test case to validate it.
- Some test cases may be designed to validate more than one testable requirement.
- Testable requirements appear to have the granularity and flexibility to make earned value a practical tool for software developers.

To redirect management focus on meeting requirements, a Systems Engineering (SE) process improvement team rewrote the SE procedures. The new procedures mandated requirements traceability and the use of technical performance measures (TPM). Per the procedure, "TPMs are used to plan and track key technical parameters throughout a development program," and "to the maximum extent practical, earned value, both planned budget and earned value taken, should be based on those TPMs that best indicate progress towards meeting the system requirements." The procedure also requires verification of the testability of

requirements.

The time-phased plan for each project phase and each build should include milestones that objectively define the functional content to be achieved at that point. The milestones should be defined in terms of incremental functionality, both the number of testable requirements and the functional capabilities to be achieved. An incremental milestone is normally defined as 100 percent of the requirements needed to achieve a functional capability. However, for earned value purposes, it can also be targeted as a lesser percentage. The functionality targets should be documented as part of the criteria for completing the milestone and taking objective earned value. The percentage target is normally related to the targeted quality, as measured by defects.

Planning for Defects and Rework

In planning for incremental builds, the Statement of Work for all builds subsequent to the first should include an estimate for rework of requirements or code to fix defects that were found in previous builds but will be fixed in subsequent builds. To ensure adequate budget and period of performance, the planning assumptions should include a planned rate or number of defects to be found in each build, and a plan to fix these defects within the rework Statement Of Work of each build. Furthermore, rework should be planned in separate work packages. Failure to establish a baseline plan for rework and to accurately measure rework progress caused many projects to get out of control.

The team's remedy was to change the EVM procedure. The procedure requires that rework is planned in separate work packages from the initial development effort and that objective metrics for rework are used for earned value.

Selection and Use of Software Metrics

For tracking progress against a plan using EVM, the most effective measures are those that address the issues, product size and stability, and schedule and progress. Three measurement categories are mapped to these issues: functional size and stability, work unit progress, and incremental capability.

The specific measures to be discussed are requirements, requirements status, component status, test status, increment content-components, and increment content-functions.

Issue: Product Size and Stability. (Category: Functional Size and Stability) The requirements measure counts the number of requirements in the system or product specification. It also counts the number of requirements that are added, modified, or deleted and provides information about the total number of requirements and the risk due to growth and/or volatility in requirements.

When incremental builds are planned, this measure is also the basic component of the measure, increment content-function, as discussed below.

Issue: Schedule and Progress. (Category: Work Unit Progress.) The recommended measures for Work Unit Progress are requirements status, component status, test status, increment content-components, and increment content-functions.

- *Requirements Status:* The requirements status measure counts the number of requirements that have been defined and allocated to software components, allocated to test cases, and successfully tested. When used to measure test status, the measure is used to evaluate whether the required functionality has been successfully demonstrated against the specified requirements. Some requirements may not be testable until late in the testing process. Others are not directly testable or may be verified by inspection.

  This measure is ideal for EVM because it is objective. The budget allocated to requirements may be equally distributed or weighted according to the estimated effort for those requirements. Consequently, requirements-based EVM, as the integrating tool for technical, schedule, and cost objectives, provides Northrop Grumman's best measure of project status, progress, and remaining effort. Since implementing requirements-based EVM, program progress has never been significantly overstated and the management control system has provided more reliable data and earlier warning of program problems (see Table 1, page 28).

  Tables 1 through 5 (see page 28) are abstracts of measures that are fully described in PSM. Per PSM, there are three aggregation structures to accumulate measurement data. Tables 1 and 2 are component-based and functional-based aggregation structures.

  Component-based aggregation structures are derived from the relationship of the system components within a particular architecture or design. For projects that implement an incremental development approach, lower-level components (such as units and configuration items) are usually mapped to the incremental delivery products as part of the aggregation structure.

  Functional-based aggregation structures define the functional decomposition of system requirements. They are often mapped to the system design components. If they are mapped to design components, then measures of the requirements (such as the number of requirements tested) can be aggregated and evaluated for a particular function.

  The data collection level describes the lowest level at which data is collected to allow problems to be isolated and understood. It can then be rolled up using the aggregation structure.
- *Component Status:* The component status measure counts the number of software components that complete a specific activity. An increase in the planned number of components may indicate unplanned growth and cost impacts. However, the number of components, although not constant, is the perpetual denominator for measuring percent complete.

  In the initial design phase for EVM, a unit of measurement should be selected based on the design standards and practices employed for each build. This may be modules, packages, pages, or another appropriate component.

  Completion of components during the design and implementation phases should be based on component reviews, inspections, walkthroughs or specified tests, as appropriate (see Table 2 page 28).
- *Test Status:* The test status measures count the number of tests attempted, executed to completion, or completed successfully. It can be applied for each unique test sequence, such as component, integration, system, and regression test and is a good basis for earned value (see Table 3, page 28).

Issue: Schedule and Progress. (Category: Incremental Capability.) Incremental capability measures count the cumulative functions or product components with a product at a given time. An increment is a predefined group of work units, functions, or product components delivered to the next phase of development. These measures determine whether the capability is being developed as scheduled or delayed to future deliveries. There are two measures of increment content.

- *Increment Content-Components:* The increment content-components measure identifies the components that are included and assembled into increments. Increment content is often deferred to preserve the scheduled delivery date. When this occurs, it is essential to quantify the deferred content in terms of earned value and to annotate the schedule to indicate that the true status and the expected completion date of the base-lined work (see Table 4).
- *Increment Content-Functions:* The increment content-functions measure is preferred for schedule progress and for earned value because it directly maps to the number of functional requirements. It requires a formal, detailed list of functions and requirements by increment, as documented in the Requirements Traceability database (see Table 5).

The completion criteria for both increment measures are successful integration and successful testing, as described in Tables 4 and 5.

Tables 1-5: *Abstracts of Measures Fully Described in PSM*

Issue: SCHEDULE AND PROGRESS **Table 1**
Aggregation Structure: FUNCTION
Category: WORK UNIT PROGRESS
Measure: REQUIREMENTS STATUS
Typically Collected for Each: REQUIREMENTS SPECIFICATION

DATA ITEM	COMPLETION CRITERIA
# Requirements Traced to: • Detailed Specifications • Software Components • Test Specifications • Tested Successfully	• Completion of Specification Review • Baselining of Specifications • Baselining Requirements Traceability Matrix • Successful Completion of all Tests, in Appropriate Test Sequence

Issue: SCHEDULE AND PROGRESS **Table 2**
Aggregation Structure: COMPONENT
Category: WORK UNIT PROGRESS
Measure: COMPONENT STATUS
Typically Collected for Each: CONFIGURATION ITEM (CI) OR EQUIVALENT

DATA ITEM	COMPLETION CRITERIA

• Total # of Components
• # of Components Completed Successfully by Activity:
  • Preliminary Design
  • Detailed Design
  • Implementation
  • Component Test
  • CI Test
	• Component Reviews, Inspections, Walkthroughs
	• Successful Completion of Specified Test
	• Released to Configuration Management
	• Resolution of Action Items

Issue: SCHEDULE AND PROGRESS **Table 3**
Aggregation Structure: SOFTWARE ACTIVITY
Category: WORK UNIT PROGRESS
Measure: TEST STATUS
Typically Collected for Each: CONFIGURATION ITEM

DATA ITEM	COMPLETION CRITERIA
• Total # Test Cases • # of Test Cases Attempted • # of Test Cases Passed	• Successful Completion of Each Test Case in Appropriate Sequence

Issue: SCHEDULE AND PROGRESS **Table 4**
Aggregation Structure: COMPONENT
Category: INCREMENTAL CAPABILITY
Measure: INCREMENT CONTENT – COMPONENTS
Typically Collected for Each: CONFIGURATION ITEM (CI) OR EQUIVALENT

DATA ITEM	COMPLETION CRITERIA
• # of Components • # of Components Successfully Integrated	• Successful Integration • Successful Testing

Issue: SCHEDULE AND PROGRESS **Table 5**
Aggregation Structure: FUNCTION
Category: INCREMENTAL CAPABILITY
Measure: INCREMENT CONTENT – FUNCTIONS
Typically Collected for Each: FUNCTION OR EQUIVALENT

DATA ITEM	COMPLETION CRITERIA
• # of Functional Requirements • # of Functional Requirements Successfully Implemented	• Successful Integration • Successful Testing

Performance–Based Earned Value

The recommended software metrics for schedule and progress are also the basis for performance-based earned value (PBEV). PBEV is a lean, cost-effective means of implementing EVM to minimize administrative costs and to focus on the big picture. It results in less work packages to track, more emphasis on objective measures of technical performance related to achieving requirements, and less emphasis on tracking support activities. PBEV has the following characteristics:

- Emphasize key performance metrics and project progress relative to plan (schedule and budget), system requirements, and TPMs that support requirements.
- Maximize budget to key technical activities.
- Measure products and product components, not tasks and inch-stones.
- Use no EV for reviews, meetings, and recurring reports.
- Manage costs, not schedule of support tasks.
- Budget for support, from the level of effort tasks can be allocated, to discrete tasks to maximize focus on technical progress.

Plan for Deferred Functionality

To prevent the overstatement of progress and the premature consumption of budget, it is recommended that the increment content-functions measure is the primary basis for earned value during the design and integration and test phases.

To illustrate how deferred functionality should be quantified at the work package level, assume that a work package for implementation of code has release of a build as its completion milestone with a budget of 500 hours. Also, assume the build includes 100 testable requirements that are budgeted to require five hours each to implement. If the build was released with 90 requirements integrated,

then earned value would be 450 hours. The event of releasing the build short of its targeted functionality is cause to close the work package and replan the remaining work. In this case, transfer the deferred requirements and the residual budget of 50 hours to the work package for the next planned build. Place the budget in the first month of the receiving work package to preserve the schedule variance. If no planned builds remain, establish them through the normal internal replan process by closing the last work package and opening a new one for the next build with the unused budget.

Process Improvement Ups Customer Satisfaction

These practices have improved our management effectiveness and increased customer satisfaction. The *Air Force Acquisition Newsletter* cited our success as follows:

> "The B-2 Spirit Stealth Bomber Program implemented several innovative process improvements using EVM. These include integrating earned value with systems engineering processes, defining improved software engineering metrics to support EVM, and developing a leaner, more effective methodology called performance-based earned value (PBEV).
>
> These changes paid off during upgrades of the B-2 weapon system. One of those upgrades was the development of the Joint Standoff Weapon/Generic Weapon Interface System (JSOW/GWIS), a software intensive effort. The new metrics helped to make it a very successful program. The PBEV methodology was used to ensure that the warfighter received the most functionality from software development efforts. On JSOW, we provided 85 percent more capability than originally planned, on schedule and under budget [3]."

The most important business objectives of a best practice are increased corporate profit and customer satisfaction. Evidence of achieving these objectives is in the Air Force quarterly assessment report of the B-2 software maintenance contract. We received excellent award fee ratings for the year ended April 2001 in all categories: technical, program management, scheduling, and cost.

Conclusion

Using earned value to plan and manage software projects can prevent expensive failures. Earned value should be based on testable requirements and selected software measures that best underlay the plan and progress to achieve all project objectives. We are now revising our systems engineering process to incorporate lessons learned and improved processes from software development.◆

References

1. Practical Software and Systems Measurement: A Foundation for Objective Project Management. U.S. Department of Defense and U.S. Army. October 2000, Version 4.0b, available at <www.psmsc.com>.
2. P. B. Wilson. "Sizing Software with Testable Requirements." Journal of Systems Development Management. August 2000, reprint available at <www.testablerequirements.com>.
3. "Aerospace Acquisition 2000." Air Force Acquisition Reform Newsletter. Jan./Feb. 2000, Vol. 3, Number 1.

About the Author

Paul J. Solomon is the director, Earned Value Management Service on the B-2 Stealth Bomber program for Northrop Grumman Corporation's Air Combat Systems, a business area of the company's Integrated Systems Sector. He is on the board of the National Defense Industrial Association, Program Management Systems Subcommittee that authored ANSI/EIA-748. He was a member of the team that received the 1998 David Packard Excellence in Acquisition Award. He presented the concepts in this article at the 2001 Software Technology Conference and the 2001 SEPG Conference in India. Solomon holds MBA and BA degrees from Dartmouth College and is a Project Management Professional.

Northrop Grumman Corporation
3520 E. Ave. M, TD21/4B
Palmdale, CA 93550
Phone: (661) 540-0618
E-mail: solompa@mail.northgrum.com