Applying Earned Value Management to Software Intensive Programs

By Robert P Hunt (Galorath Incorporated), Paul J. Solomon (Performance-Based Earned Value), and Dan Galorath (Galorath Incorporated)

Often, traditional earned value approaches do not deal sufficiently with the idiosyncrasies of software intensive programs. However, successful management of software intensive programs can be achieved by focusing on establishing the requirements, developing a reliable baseline estimate for cost and schedule, selecting effective software metrics, applying Performance-Based Earned Value® (PBEV), *and using analytic processes to project cost and schedule based on actual performance.*

Introduction

The Department of Defense estimates that software now accounts for 40% of all research, development, test and evaluation (RDT&E) spending[i]. Software intensive projects that achieve their original cost and schedule projections are rare. Many information technology projects have been declared a disaster area by commercial and government managers. These projects have been too costly, too late, and often don't work right. Applying appropriate technical and management techniques can significantly improve the current situation.

Inaccurate estimates can threaten project success causing poor project implementations, the shortcutting critical processes, and emergency staffing to recover schedule. The lack of well defined project requirement and specifications may result in significant growth in cost and schedule. Symptoms of this growth may include constantly changing project goals, frustration, customer dissatisfaction, cost overruns, missed schedules, and the failure of a project to meet its objectives.

PMI published an analysis of several government defense and intelligence agency large-scale acquisition programs that experienced significant cost and schedule growth. This analysis shows that several critical factors need to be addressed in the pre-acquisition phase of the acquisition cycle. The principal causes of growth on these large-scale programs can be traced to several causes related to overzealous advocacy, immature technology, lack of corporate technology roadmaps, requirements instability, ineffective acquisition strategy, unrealistic program baselines, inadequate systems engineering, and work-force issues.[ii] This paper will discuss some key element associated with:

- Establishing a process for requirements definition and developing the cost and schedule baseline
- Developing a reliable cost and schedule baseline,
- Identifying critical software management metrics,
- Applying Performance-Based Earned Value (PBEV), and
- Using an analytic process (such as SEER Control; formerly called Parametric Project Monitoring and Control (PPMC)) to project cost and schedule based on actual performance.

Establishing a Process for Requirements Definition and Developing the Technical, Cost and Schedule Baselines

A software program life cycle cost estimate is the most knowledgeable statement one can make at a particular point in time regarding effort/cost, schedule, staffing, risk, and reliability. However, the most important business decisions about a software project are often made at the time of minimum knowledge and maximum uncertainty. Cost estimators recognize that the estimate is not a point, but rather a well formed estimate defined by a probability distribution.

A well defined process is critical to defining the requirements and completing the initial cost and schedule estimate. The proper use of PBEV provides for integration of project technical scope, schedule, and cost objectives; and the establishment of a baseline plan for performance measurement. Additionally, the use of an analytic tool to project likely cost and schedule based on actual performance provides for realistic projections of future performance. Success of the project can be aided by defining the best objectives, by planning resources and costs which are directly related to those objectives, by measuring accomplishments objectively against the plan, by identifying performance trends and problems as early as possible, and by taking timely corrective actions.

A CMMI tutorial recognizes that people, process, and technology are major determinants of product cost, schedule, and quality. We all realize the importance of having a motivated, quality work force but even our finest people can't perform at their best when the process is not understood or not operating at its best. Figure 1, People, Process, Technology are Keys, presents this concept.

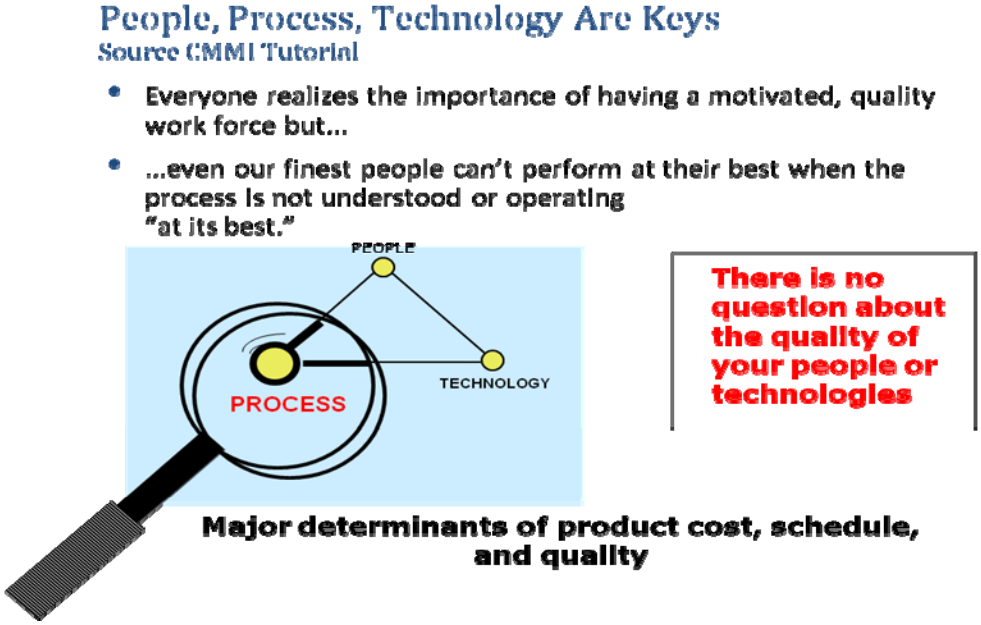

Figure 1, People, Process, Technology are Keys

In the book, “Software Sizing, Estimation and Risk Management” (Dan Galorath and Michael Evans, 2007) a ten step process is presented for program requirements generations and estimation. Figure 2, 10 Step Software Process, outlines the ten steps.

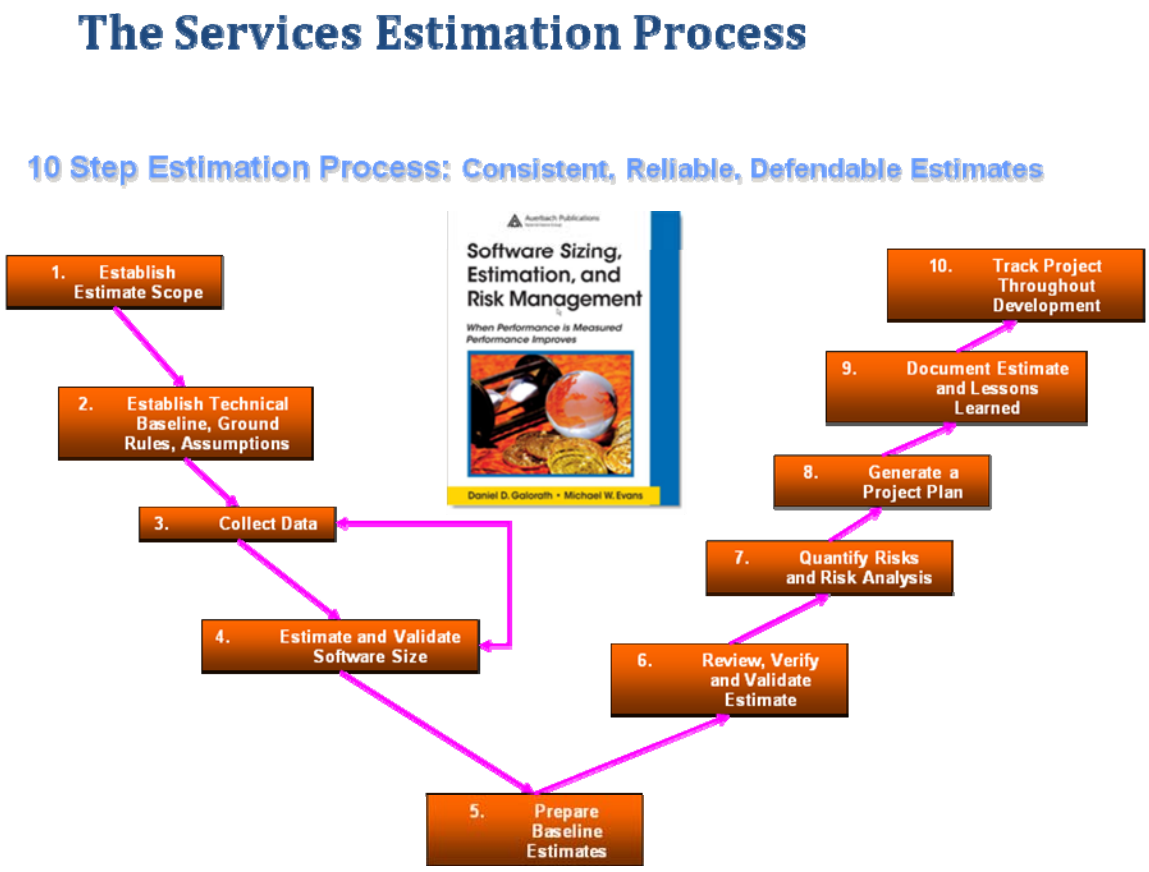

Figure 2, 10 Step Software Process

While the Galorath process includes ten steps, other process may include more or less steps (e.g. the GAO Cost Guide includes a 12 steps process). Note specifically the importance of step 4, *estimating and validating the software size metric*. The key here is to establish an auditable, repeatable set of steps to establish the requirements and develop the baseline estimate of cost and schedule. This is the key to articulating an accurate requirement and establishing a reliable baseline for cost and schedule.

Identifying critical software management metrics

That most large software programs get into trouble is a demonstrated phenomenon. Therefore selecting the correct set of software metrics to track is critical to program success. Practical Software Measurement (McGarry, Card, Jones; Addison-Wesley, 2002) identifies seven information categories and then expands these information categories into measurable concepts and then prospective metrics[iii]. This taxonomy is presented in the Figure 3, What To Measure.

		WHAT TO MEASURE	
			Information Category Measure Mapping*
	Information Category	Measurable Concepts	Prospective Measures
1	Schedule and Progress	Milestone completion	Mileston Dates
		Critical Path Performance	Slack Time
		Work Unit Progress	Requirements Traced, Requirements Tested, Problem Reports Opened, Problem Reports Closed, Reviews Completed, Change Requests Opened, Change Requests Resolved, Units Desgined, Units Coded, Units Integrated, Test Cases Attempted, Test Cases Passed, Action Items Opened, Action Items Completed
		Incremental Capacity	Components Integrated, Functionality Integrated
2	Resources and Cost	Pewrsonnel Effort	Staff Level, Development Effort, Expereince Level, Staff Turnover
		Financial	BCWS, BCWP, ACWP, Budget, Cost
		Environmental/Support	Qualaity Needed, Quality Available, Time Available, Time Used
3	Product Size & Stability	Physical Size/Stability	Database Size, Compomnents, Interfaces, LOC
		Funtional Size	Requirements, Function Changes, Function Points
4	Product Quality	Functional Correctness	Defects, Age of Defects, Technical Performanmce
		Maintaniability	Time to Release, Cyclomatic Complexity
		Efficeincy	Utilization, Throughput, Response Time
		Portability	Stand Comp-Oliance
		Usability	Operator Errors
		Realibility	MTTF
5	Process Performance	Process Cxompliance	Reference Maturity Rating, Process Audit Findings
		Process Efficiency	Productivity, Cycle Time
		Process Effectiveness	Defects Contained, Defects Escaping, Rework Effort, Rework Components
6	Technology Effectiveness	Technology Suitability	Requirements Coverage
		Technology Volatility	Baseline Changes
7	Customer Satisfaction	Customer Feedback	Satisfaction Rating, Award Fee
		Customer Support	Request for Support, Support Time
			* Practical Software Measurement; McGarry, Card, Jones; Addison-Wesley2002

Figure 3, What To Measure

Collecting and tracking data on all the prospective metrics is impractical for typical software intensive programs. Software developers often produce their software deliverables in unique environments and with unique processes. In selecting the appropriate software metrics, the analyst must "do your own thing, but carefully." [iv]

For Earned Value purposes, the most effective software metrics are those that relate to product size, schedule, quality, and progress. For software intensive programs, measures of quantity (e.g. number of lines of code completed) do not accurately reflect the quality aspects of the work performed on neither the program nor the actual progress since items such as lines of code completed do not capture items such as integration, testing, etc.

Size is often measured as Source Lines of Code (SLOC) or Function Points and used as a sizing measure for budgets and for earned value using a percent of completion method. There are two critical problems with this approach. First, there has traditionally been a significant error in estimating SLOC. And, the number of lines of code completed does not necessarily reflect the quality or total progress toward a performance goal. Therefore, any progress metric based solely on SLOC is highly volatile. Whether SLOC, function points, Use Cases, or some other size artifact is selected, a careful process must be utilized to establish a credible size metric. It is recommended that in addition to tracking progress toward a goal, size growth should also be tracked.

Schedule metrics and procedures normally relate to completion milestones are also a common tracking metric. Sometimes these milestone definitions and completion criteria lack quantifiable objectives. Often an incremental build is released that does not incorporate all the planned functional requirements or a developer claims victory after just testing the nominal cases.

Progress metrics can be very difficult for large software programs. It is generally agreed that no software is delivered defect free. Software engineers have hoped that new languages and new processes would greatly reduce the number of delivered defects. However, this has not been the case. Software is still delivered with a significant number of defects. Capers Jones estimates that there are about 5 bugs per Function Point created during Development[v]. Watts Humphrey found "… even experienced software engineers normally inject 100 or more defects per KSLOC[vi]. Capers Jones says, "A series of studies the defect density of software ranges from 49.5 to 94.5 errors per thousand lines of code[vii]." The physical and practical limitations of software testing (the only way to determine if a program will work is to write the code and run it) ensure that large programs will be released with undetected errors. Therefore, defects discovery and removal is a key metric for assessing program quality.

The analyst should review the list of potential measures defined in Figure 3, What To Measure, and select the set of metrics that are most appropriate for a specific program.

Applying Performance-Based Earned Value (PBEV)

Performance-Based Earned Value® (PBEV) is an enhancement to the Earned Value Management Systems (EVMS) standard[viii]. PBEV overcomes the standard's shortcomings with regard to measuring technical performance and quality (quality gap). PBEV is based on standards and models for systems engineering, software engineering, and project management that emphasize quality. The distinguishing feature of PBEV is its focus on the customer requirements. PBEV provides principles and guidance for cost effective processes that specify the most effective measures of cost, schedule, and product quality performance.

Program managers expect accurate reporting of integrated cost, schedule, and technical performance when the supplier's EVMS procedure complies with the EVMS Standard. However, EVM data will be reliable and accurate only if the following occurs:

- The indicated quality of the evolving product is measured.
- The right base measures of technical performance are selected.
- Progress is objectively assessed.

Using EVM also incurs significant costs. However, if you are measuring the wrong things or not measuring the right way, than EVM may be more costly to administer and may provide less management value[ix].

Because of the quality gap in the EVMS standard, there is no assurance the reported earned value (EV) is based on product metrics and on the evolving product quality. First, the EVMS standard states that EV is a measurement of the quantity of work accomplished and that the quality and technical content of work performed are controlled by other processes. A software manager should ensure that EV is also a measurement of the product quality and technical maturity of the evolving work products instead of just the quantity of work accomplished. Second, the EVMS principles address only the project work scope. EVMS ignores the product scope and product requirements. Third, the EVMS standard does not require precise, quantifiable measures of progress. It states

that objective EV methods are *preferred* but it also states that management assessment (subjective) may be used. In contrast, other standards specify objective measurement. Fourth, EVM is perceived to be a risk management tool. However, EVMS was not designed to manage risk and provides no guidance on the subject.

PBEV is a set of principles and guidelines that specify the most effective measures of cost, schedule, and product quality performance. It has several characteristics that distinguish it from traditional EVMS:

- Plan is driven by product quality requirements.
- Focuses on technical maturity and quality, in addition to work.
- Focuses on progress toward meeting success criteria of technical reviews.
- Adheres to standards and models for systems engineering, software engineering, and project management.
- Provides smart work package planning.
- Enables insightful variance analysis.
- Ensures a lean and cost-effective approach.
- Enables scalable scope and complexity depending on risk.
- Integrates risk management activities with the performance measurement baseline.
- Integrates risk management outcomes with the Estimate at Completion.

PBEV augments EVMS with four additional principles and 16 additional guidelines.

The following are PBEV principles that set it apart from EVMS:

1. Product Scope and Quality. Integrate product scope and quality requirements into the performance measurement baseline.
2. Measure Quality. Specify performance toward satisfying product quality requirements as a base measure of earned value.
3. Integrate Risk. Integrate risk management with EVM.
4. Tailored Application. Tailor the application of PBEV according to the risk.

Figure 4, EVMS and PBEV Flow Chart, overlays the "Performance- Based" process flow on the traditional "Earned Value" process flow.[x]

PBEV supplements traditional EVMS with the best practices. Its principles and guidelines enable true integration of project cost, schedule, and technical performance. The distinguishing feature of PBEV is its focus on the customer requirements. Measures of product scope and product quality are incorporated into the project plan. Progress is measured against a plan to fulfill all customer requirements. Measuring the wrong things does not dilute management attention. Consequently, management is able to take rapid corrective actions on deviations that threaten customer satisfaction and business enterprise objectives.

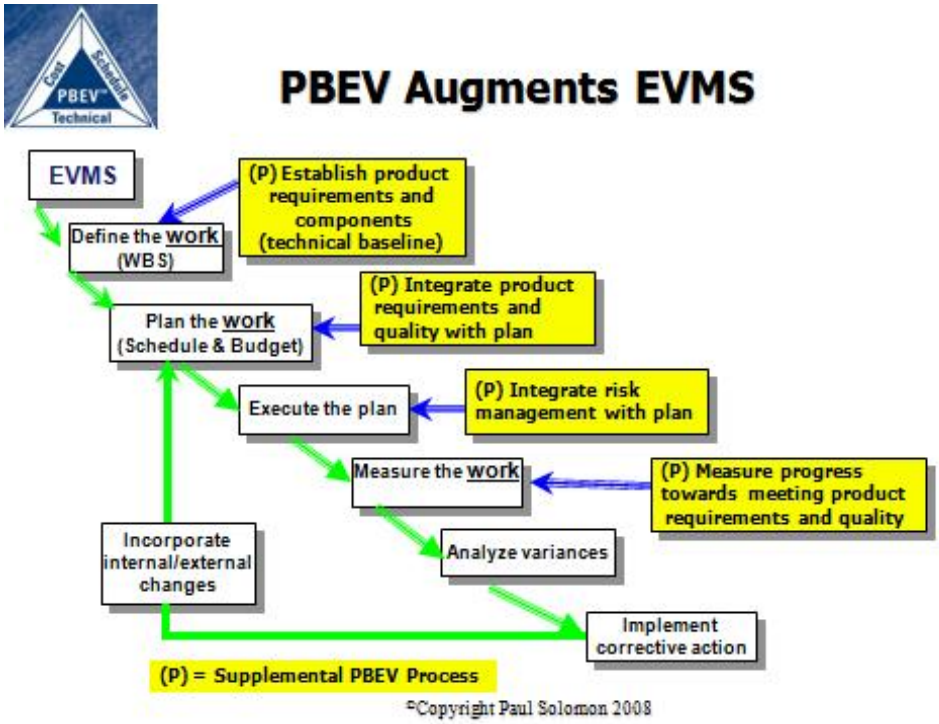

Figure 4, EVMS and PBEV Flow Chart

Using an analytic process to project cost and schedule based on actual performance

Once the requirement definition is complete; the cost and schedule baseline has been established; the appropriate metrics have been selected; and a PBEV system is in place, the final challenge is to implement a process that quickly and accurately estimates final cost and schedule based on actual performance. This analysis is best accomplished using an analytic/parametric process. Galorath Incorporated calls this process SEER Control (formerly Parametric Project Management and Control. The purpose of SEER Control is to provide an understanding of the project's progress so that appropriate corrective actions can be taken when the project's performance deviates significantly from the plan. SEER Control applies a four-dimensional (4-D) approach for assigning progress to the development of each program/application that is part of the project. The first dimension is **Software Development Life Cycle (SDLC) primary activity completion** for the development of a specific program/application. Each SDLC primary activity, in turn, is assigned progress according to a weighted combination of three other dimensions: **artifact completion**, **milestone completion**, and **defect discovery/removal**. SEER Control provides an "at-a-glance" indication of project status. This concept is presented in Figure 5, Understanding and Tracking Defects and Other Metrics. This analytic process uses actual performance to re-estimate the anticipated cost and schedule. The "dashboard" at the bottom of Figure 5, presents a health and status indicator for the project. In Figure 5, five metrics are tracked, schedule variance, time variance, cost variance, size growth, and defects discovery and removal. SEER Control allows you to track size growth and actual defect metrics. Size growth can indicate growth in

requirements and can be an indicator of why a project may be off track. The profile of defects reported and removed is compared against the estimated time phased defects.

In Defects Tracking, the analyst will see the estimated defects reported and actual defects reported. When reported defects are lagging the estimated defects, that could indicate that not enough testing is being performed, especially if the actual defects removed are tracking with the estimated defects removed. Conversely, if the actual defects removed lag the estimated, but defects reported tracks with the estimated, then you may not have enough programming resources to make fixes. If actual defects reported and removed follow the general profile of the estimated, but are higher or lower, then the baseline project estimate may be over or underestimated. SEER Control also tracks the Time Variance (TV). The TV is the cumulative difference in schedule months between earned value and the baseline plan up to the date of the latest snapshot. When a rollup element is selected, the time variance is equal to the worst time variance of its subordinate programs. Positive values are favorable, negative values are unfavorable. SEER Control also tracks the Time Performance Index (TPI). The TPI is the time efficiency achieved from the beginning of development to the date of the latest snapshot. A performance index greater than one is favorable. A performance index less than one is unfavorable. The Time Performance Index (TPI) is the ratio of the elapsed time from the Actual Start Date to the baseline planned date and the elapsed time from the Actual Start Date to the snapshot date.

Other metrics can be tracked. In addition to the health and status indicator using the red, yellow, green indicators, this automated application re-baselines the program estimate to present a revised cost and schedule prediction.

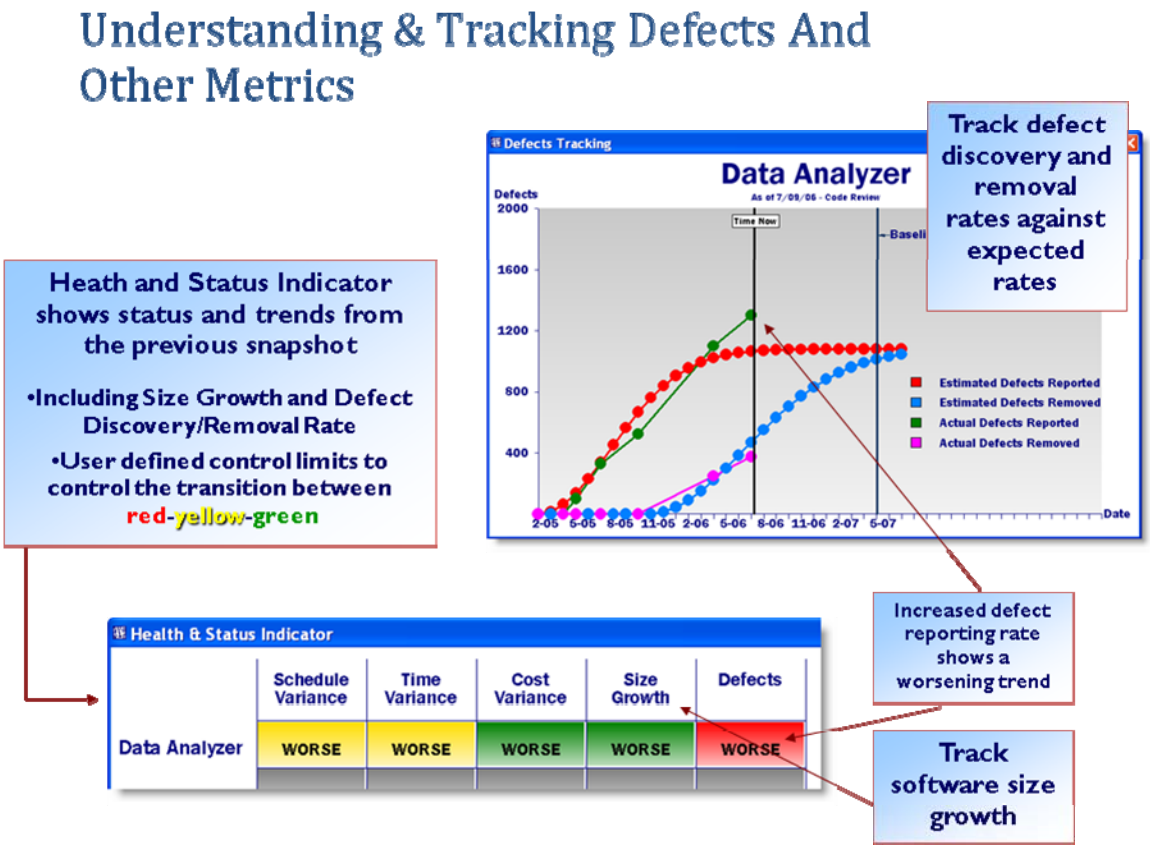

Figure 5, Understanding and Tracking Defects and Other Metrics

At the heart of the PPMC vision is the desire to forecast the final project outcome based on performance to date. One of the primary goals of PPMC is to provide adequate supporting documentation (charts and reports) to support the software project management process and to satisfy stakeholder needs.

Conclusion

Using earned value to plan and manage software intensive projects can prevent expensive failures. Earned value should be based on the foundation of establishing the requirements, developing a reliable baseline estimate for cost and schedule, selecting effective software metrics, applying Performance-Based Earned Value (PBEV), and using analytic processes to project cost and schedule based on actual performance.

Authors Biographies

Bob Hunt is the Vice President, Services of Galorath Incorporated. As Vice President for Services, Mr. Hunt is responsible for the management and technical direction of the services staff and the quality of the services products. Galorath's professional services staff is defined by expert analysts and consultants with broad experience in cost estimating and project management, as well as deep knowledge of the capabilities, features and limitations of SEER tools. Mr. Hunt has provided software program assessments, SEI Checklist evaluations, software sizing analyses, and

software cost estimating for commercial and federal clients including the Customs and Border Patrol, the Department of Defense, NASA, and various commercial clients. Prior to joining Galorath, Mr. Hunt was President of CALIBRE Services, Inc. a subsidiary of CALIBRE Systems, Inc. Prior to joining CALIBRE, he was a Vice President of Science Applications International Corporation (SAIC) responsible for the Cost and Acquisition Management Operation. As a civil servant, Mr Hunt was Deputy Director of Cost Analysis for Automation and Modeling, Cost Analysis Division, U.S. Army, The Pentagon. In this position, Mr. Hunt was instrumental in setting up and developing the U.S. Army Cost and Economic Analysis Center and was the principal author of the initial Army Cost Analysis Manual. Mr. Hunt has held leadership positions and made technical presentations for the American Institute of Aeronautics and Astronautics (AIAA), the Society of Cost Estimating and Analysis (SCEA), and the National Association of Environmental Professionals (NAEP), and the IEEE.

Paul Solomon, PMP is the co-author of the book, Performance-Based Earned Value®. He is internationally recognized as a leader, teacher, and consultant on Earned Value Management (EVM). He published many articles on EVM, systems engineering, software engineering, and risk management. Most are available on his website, www.PB-EV.com. He retired from Northrop Grumman Corporation where he led the use of EVM on programs including the B-2 Stealth Bomber, Global Hawk, and F-35 Joint Strike Fighter. He has taught thousands of professionals and led EVMS implementation, compliance reviews, Integrated Baseline Reviews, independent assessment reviews, and process improvement teams. He is qualified to lead EVMS certification reviews.

Daniel D. Galorath has over three decades in the software industry. Daniel D. Galorath has been solving a variety of management, costing, systems, and software problems for both information technology and embedded systems. He has performed all aspects of software development and software management. One of his strengths has been reorganizing troubled software projects, assessing their progress applying methodology and plans for completion and estimated cost to complete. He has personally managed some of these projects to successful completion. He has created and implemented software management policies, and reorganized (as well as designed and managed) development projects. His company, Galorath Incorporated, has developed the SEER applications, methods, and training for 1) software, 2) hardware, electronics & Systems, 3) Information Technology, and 4) Manufacturing cost, schedule, risk analysis, and management decision support. He is one of the principal developers of the SEER-SEM™ software evaluation model. His teaching experience includes development and presentation of courses in Software Cost, Schedule, and Risk Analysis; Software Management; Software Engineering; to name a few. Mr. Galorath is a sought after speaker. Among Mr. Galorath's published works are papers encompassing software cost modeling, testing theory, software life cycle error prediction and reduction, and software and systems requirements definition. Mr. Galorath was named winner of the 2001 International Society of Parametric Analysts (ISPA) Freiman Award, lifetime achievement award, awarded to individuals who have made outstanding contributions to the theoretical or applied aspects of parametric modeling. Mr Galorath's book "Software Sizing, Estimation, and Risk Management" was published March 2006. Mr. Galorath publishes a blog, "Dan Galorath on Estimating" at http://www.galorath.com/wp/.

[i] Page 134, Trillions For Military Technology; John A.Alic, Palgrave MacMillian, 2007

[ii] PMI Project Management Journal, March 2008; "Best Project Management and Systems Engineering Practices in the Preacquisition Phase for Federal Intelligence and Defense Agencies" by Steven R. Meier
[iii] Page 37, Practical Software Measurement; McGarry, Card, Jones; Addison-Wesley 2002
[iv] P. 68, A Practical Guide to Earned Value Project Management, Charles Budd and Charlene Budd; Management Concepts, 2005
[v] Geriatric Issues of Aging Software, Capers Jones, CrossTalk, Dec 2007, Vol. 20 No 12
[vi] Humphrey, W.,"A Personal Commitment to Software Quality." Pittsburg, PA: The Software Engineering Institute (SEI)
[vii] Jones, T.C. Programming Productivity. New York: McGraw-Hill, 1972
[viii] American National Standards Institute. "Earned Value Management Systems." (ANSI)/EIA-748-A-1998. Apr. 1998. Reaffirmed 28 Aug. 2002.
[ix] Solomon, Paul J. "Integrating Systems Engineering With Earned Value Management." PMI Measurable News, Fall, 2008 and Defense AT&L May/June 2004:42 Links at www.PB-EV.com, Advanced EV:PBEV tab..
[x] Page 17, Solomon, Paul and Young, Ralph, "Performance-Based Earned Value®"Hoboken, NJ:Wiley & Sons, 2007.