Analysis and Application of Earned Value Management in Software Development

Frank Tsui
School of Computing and Software Engineering
Southern Polytechnic State University, Marietta, Georgia, USA 30060

Abstract - *Earned Value Management (EVM) is a well-known cost and schedule management technique in government and defense industry projects. Its usage, however, is not as wide spread in the general software industry. In this paper we explore some of the shortcomings in EVM and suggest several improvements in the application of EVM in software development projects. In particular, we analyze the nature of software projects and offer improved ways to approximate Estimate of Completion for cost of software projects. Another area of special attention in this paper is demonstrating that EVM's metric such as Schedule Performance Index alone does not alert the project managers early enough on late task starting date and that the actual start date itself must be taken into account.*

Keyword: EVM, Cost, Schedule, Software Projects

1 Introduction

Earned Value Management (EVM) is a cost-schedule management and control technique. Since its inception in government financial management approximately fifty years ago, EVM has been used for project management in various government and defense industry related projects [1, 6, 7]. However, its usage is still relatively new in the software industry [5]. EVM has been adopted by some Agile software development projects [8], but it remains foreign to a large number of software project managers. In this paper we explore and analyze the applicability of this technique as it applies to software project management and to software engineering. Through this analysis, we will delineate some of the potential drawbacks in handling projections and show how one may cope with these in order to incorporate EVM as a project management technique for software engineering. In particular, at the macro level, we offer potential improvements to Estimate of Completion (EAC) for cost. We also offer suggestions at the micro level where examining only the numerical metrics such as Schedule Performance Index (SPI) and Cost Performance Index (CPI) of EVM alone may not be enough. We show that while EVM metrics are mostly task completion driven, project managers still need to guard against delay in task start date.

In the next section, Macro Level Definition and Application of EVM, we quickly review the familiar terms and the specific metrics employed in EVM. We then explain the forecasting problem of Estimate at Completion (EAC) as it is applied to software development projects. Some possible ways to improve on the EAC forecasting problem is offered in the Potential Improvements section. In the Micro Level Application of EVM section, we discuss ways to account for and to provide credits to tasks that have started and have completed part of the tasks. The need to be mindful of slippage in task start date and having to look beyond SPI and CPI metrics is discussed in Task Late Start Problem section. Lastly, we summarize the analysis of application of EVM to software projects in the Conclusion section.

2 Macro Level Definitions and Application of EVM

At the macro level, all large software development projects have a similar set of major activities: requirements analysis and specification,

architectural design, detailed design, code implementation, build and integration, functional testing, system testing, and release for deployment. In a way, one may view this as the work breakdown structure (WBS) of macro tasks in software development [9]. For each of these macro tasks one may assign it an estimated amount of effort, or a Budgeted Cost (BC), along with an estimated start date and completion date. The sum of all the allocated cost of these tasks, or ∑BC, at project completion is called Budget At Completion (BAC). The major elements of EVM are composed of Budgeted Cost of Work Scheduled (BCWS), Budgeted Cost of Work Performed (BCWP), and Actual Cost of Work Performed (ACWP). At any project status check time, t, the sum of BCs of all those tasks that are scheduled to be completed is BCWS. For those tasks that are actually completed or performed at time t, the sum of those tasks' BCs is the BCWP. Both BCWS and BCWP use the estimated work effort, BCs, but they may contain different set of tasks because scheduled and actually completed tasks at time t may be different. The sum of the Actual Effort (AC) expended for those actually completed tasks at time t is ACWP. Thus, at time t, ACWP is the sum of ACs for those same tasks accounted for in BCWP.

Furthermore, one may now track and compare the planned versus the actual project status using a pair of measurements, Schedule Variance (SV) and Cost Variance (CV). An alternate, but similar, pair of Schedule Performance Index (SPI) and Cost Performance Index (CPI) may be used to monitor the project status and provide an indicator of how the project is performing relative to its planned schedule and cost. Note that SPI = BCWP/BCWS and CPI = BCWP/ACWP; thus when SPI > 1, we are ahead of schedule, but when CPI > 1, we are under-running the cost. When SPI and CPI are both equal to 1, then we are on target for both schedule and cost. When SPI < 1, we are behind schedule, but when CPI < 1 we have a cost over-run situation. As a project manager, one may ask for guidance on how much smaller than 1 does SPI need to be before one should be alarmed. Similarly, how much smaller than 1 does CPI needs to be before one should feel uneasy? These may be industry or enterprise specific, and there is no general guidance at this time.

EVM also includes a computational formula for prognosticating Estimate at Completion (EAC) for cost and for schedule as follows. Let EAC_S stand for EAC for Schedule and EAC_C stand for EAC for Cost. Then,

$$EAC_S = ACWP + (BAC - BCWP)/SPI \quad (1)$$
$$EAC_C = ACWP + (BAC – BCWP)/CPI \quad (2)$$

These forecasted numbers are based on two components. One part is the actual cost incurred to-date, or ACWP, and the other part is the estimated remaining work, or (BAC - BCWP). A vital part of estimating the remaining work utilizes a feed-back mechanism of in-project history, SPI and CPI, for estimating respectively the new schedule at project completion and the new cost at project completion. This method of predicting EAC has been shown to be relatively accurate for some projects [3]. An earlier study of differing EAC by varying the CPI and SPI indices was conducted by Christensen [4] and found that one still need to be cautious in EAC projection.

2.1 Forecasting Problem for Software Projects

For software development, especially for large software projects, the major tasks involving requirements, design, implementation, etc. are very different in nature and are often times performed by different people with varying levels of skills. For example, requirements solicitation and analysis work is very different from implementation work. It is also very different from design or testing tasks. These macro tasks in software development are each composed of different activities and thus require different skills sets. The different tasks of software development would present at least the following list of differences that will make productivity and actual performance information of any one major task practically non-applicable to another major task of software development for in-project feedback.

- Different set of sub-activities and skills required for any major task
- Different sets of people performing that major task
- Quality effects of one task on later tasks

For example, as we progress from completing the requirements task to the other tasks, the in-project history of requirements task for that project may not be appropriate for estimating the remaining effort, both cost and schedule, of implementation task or of testing task of the project. Thus using in-project SPI

and CPI indices, at the completion of requirements task may actually be misleading in projecting task completions for other tasks because the other tasks are significantly different in nature. Therefore, the resulting EAC_S and EAC_C may create false hope or false alarm.

The quality effect in software development is an example that relates to the special situation of software development that EVM does not account for. Because software development tasks such as design, implementation and even testing still include many creative inventions, the likelihood of errors is high. The projection of defects from these errors, which must be corrected prior to release, is not very accurate. Thus the amount of extra work needed in down-stream activities, such as different testing and bug fixing tasks, resulting from these potential errors in earlier, upstream activities, can not be folded into the CPI or SPI indices. Thus EAC projections for schedule and cost, which do not take this quality effect into consideration, may be off the mark for software development projects. This notion of software development often taking longer than expected is also highlighted in [2].

2.2 Potential Improvements

We propose two categories of approaches to improve the projection for software projects. First category is to address the situation via using a different set of feedback mechanism than the generic in-project CPI and SPI. We need to recognize that each type of task in software development is markedly different from other types of tasks. The historical information relating to each different type of remaining task should be used as the new cost and schedule adjuster. This assumes an organization that either has historical data or has access to historical data. Let the remaining major tasks be E1, ---, En. Corresponding to each E_j, we will define BC_j, and AC_j to be the Budgeted Cost and Actual Cost of those efforts related only to that major task which is still remaining. Then define a $CPI_j = BC_j/AC_j$ from past data for each remaining major task j. Note that we use BC and AC of each task because we are interested in delineating each specific task. In the most simplified case, we would just use the average of these CPI_j 's as the new CPI'. Thus for the remaining n tasks, the new CPI' = ($\sum CPI_j$)/ n. The new Estimate at Completion for Cost will then be the following.

$$EAC_C' = ACWP + (BAC - BCWP)/ CPI' \quad (3)$$

Such a derived CPI' may be still too simplified. We may also look at each CPI_j for the remaining tasks and consider the following:

$$CPI'_{max} = \text{maximum } (CPI_j s) \quad (4)$$
$$CPI'_{min} = \text{minimum } (CPI_j s) \quad (5)$$
$$CPI'_w = w1xCPI_{j1} + w2xCPI_{j2} + ----+ wnCPI_{jn}, \text{ where } 0<wi<1 \text{ and } \sum wi = 1 \quad (6)$$

CPI'_{max} is the CPI' derived from the type of task whose historical BC/AC ratio is the largest among all the remaining tasks. Thus CPI'_{max} may be considered for an optimistic prognostication of EAC. CPI'_{min} is the reverse case and may be considered for a pessimistic prognostication. Finally, the project manager may place different weights on each of the remaining tasks' CPIs and create a weighted CPI', or CPI'_w. Using past projects' information closely related to the remaining tasks, when picking the appropriate CPI', brings an additional level of accuracy in the projections. The software project manager may consider each of the following for his or her Estimate at Completion:

$$EACc'1 = ACWP + (BAC\text{-}BCWP)/CPI'_{max} \quad (7)$$
$$EACc'2 = ACWP + (BAC\text{-}BCWP)/CPI'_{min} \quad (8)$$
$$EACc'3 = ACWP + (BAC\text{-}BCWP)/CPI'_w \quad (9)$$

The application of weights utilized in CPI'w may now take quality results of earlier tasks into consideration for the yet to finish tasks. Software project managers may also bring in other risk considerations in choosing which of the above projections to use. Aforementioned quality effects may be folded in as an additional risk consideration. If it is known from historical data that certain tasks such as design or implementation is more error prone for the particular set of developers of the project, then subsequent testing and fix tasks may require more effort than projected. Thus the project manager may choose to be more conservative and use EACc'2 for estimating the completion cost. EACc'3 may be most complicated, but it is the most flexible in that it allows the project manager to allocate different levels of concern for each of the different types of remaining tasks. Note that EACc' is just a special case of EACc'3 where the weights in CPI'w are all equal.

The above discussion is based on enterprises that have collected and kept historical data on software development. Moreover, these enterprises need to

have monitored projects using EVM and kept the BCs and ACs for the major software development tasks. However, most of the enterprises have not used EVM nor kept such data related to past BC and AC by major task types. Hence, a different approach is needed for those enterprises who are new to EVM and do not have historical data. These organizations would still need to use the in-project data to estimate the remaining efforts for EAC.

A different category of approach to improving the projection of EAC is to modify the project management approach and choose the appropriate projects to fit the measurement, not modifying the projection formula. When the project is small, we often times use a small group of people of similar skills and have these same people involved in all aspects of the project. This management approach for small projects is sometimes a result of economic practicality where specialization by task type is just not possible. Such an approach will take out one of the earlier mentioned problems of different specialists performing different tasks and thus causing inaccurate prognostication of remaining tasks based on in-project data. Furthermore, apply EVM to not only small, but also simpler projects where less new innovation is required. This will ease the potential quality variance problem. Limiting the application of EVM to small, simple software project and employing the same set of people to perform all major tasks, should provide more stability to in-project data for estimating even different tasks. This approach should provide a partial improvement to using the original formulae of EAC_S and EAC_C.

3 Micro Level Application of EVM

Adoption of EVM for software engineering also needs some guidance at the micro level. One difficult area is the computation of BCWP. For project status analysis and report at time t, we accumulate all the BC's of the tasks that are completed by time t. That becomes the BCWP at time t and is used for CPI and SPI indices. However, at time t there may be tasks that are partially complete. Traditionally, these partially completed tasks are not included in the computation of BCWP. In other words, a task gets either 0% or 100% of the BC when computing the BCWP. In software development many tasks are performed in parallel, and these partially completed tasks need to be folded in the computation of BCWP to gain a more accurate account of the project status. Consider Figure 1 where multiple tasks are performed in parallel. The project of status of BCWP on March 15 would include those tasks that are completed, namely, requirement and design. Thus BCWP would be 100 + 76 = 176 person hours, ACWP would be 105 +65 = 170 person hours and BCWS would be 100 + 76 = 176 on the March 15 status report. The schedule performance index, SPI = BCWP/BCWS = 176/176, would be 1, and cost performance index, CPI = BCWP/ACWP = 176/170, would be greater than 1. This says that we are *right on schedule* and *under-run on cost*. But we also need to account for the other two major tasks, implementation and testing, that have started and have already expended some effort in our status report. Furthermore, we note that the testing activity actually started later than the estimated start date.

Figure 1: Project Status on March 15

Major Task	Estimated Effort (BC) In person hrs.	Actual Effort Expended (AC) In person hrs.	Estimated Start date	Actual Start date	Estimated Completion date	Actual Completion date
Requirement	100	105	Jan 10	Jan 10	Feb 15	Feb 15
Design	76	65	Feb 10	Feb 10	March 10	March 5
Implement	150	55	March 1	March 1	April 15	-----
Test	90	35	Feb 10	Feb 15	May 1	-----
Integrate	8	0	May 2	-----	May 5	-----

Several approaches may be taken. If the software development project is laid out in broad chunks of requirements, design, implementation, test, integration and release, we can see that these tasks will overlap. For example, that part of testing which addresses test scenario and test case development may overlap with requirements and design tasks. Taking a project status at the end of the requirements phase and not allocate any credit to some sub-task completions within testing would create a false impression of the status. To evade this type of problem, partial task completion credit may be given with rules such as the following.

- Allocate 0% of the BC of the task if it has started but not reached 30% of the task BC.
- Allocate 25% of the BC of the task if it is started and has passed 30% of the task BC.
- Allocate 100% of the BC of the task only when it is completed.

Including the tasks' partial BCs and adding that into the computation of BCWP may provide a more

accurate account of the project status. The above suggestion is a relatively conservative approach. Using this conservative credit allocation scheme, the new BCWP = 100 + 76 + (.25 x 150) + (.25 x 90) = 236. The new ACWP = 105 + 65 + 55 + 35 = 260. Utilizing the new BCWP and ACWP, we can recalculate for new SPI = 236/176 and new CPI = 236/260. The new SPI > 1, and the new CPI < 1; thus with partial credits, our project status on March 15 would be quite different from what was previously stated. We are now *ahead of schedule*, but has a slight *cost over-run*.

Clearly, one may vary the 25% to something lesser or larger, depending on past experiences and history with the project team and the nature of the current project. One may also add more increments and granularities such as allocating only 10% when the task is started, allocating 25% when the task is believed to be half way completed, and allocating 50% when it is believed to have past the half way mark. Since the adoption of EVM in software development has not been broad enough, no clear guidance on the percentage can be provided yet. However, we do know the binary case of 0% or 100% creates some problems projecting an accurate project status when there is more than one task that has already incurred some effort, including those tasks that started earlier than planned.

3.1 Task Late Start Problem

If a task is started earlier than planned, we can adopt some variation of the above suggested, partial credit approach. However, in our Figure 1 example, the testing activity actually started five days later than the planned start date of February 10th. While one can easily see this delay in test starting date in a tabular form such as Figure 1, there is no clear way to indicate this delay in task starting if one just looked at EVM metrics of SPI and CPI. The previously computed project SPI indicated that the project was on schedule on March 15. The newly computed SPI, with partial credits, even indicated that the project was ahead of schedule. Experienced software engineers and project managers know that a late starting task often results in late completion of that task and/or requires more effort and may even adversely affect other related tasks. Project managers may wonder whether they should have been alarmed on March 15 by the start date delay. Let us fast forward from March 15 status day to May 1, when the testing task is scheduled to compete. To keep the discussion focused, we will assume that the implementation task actually completed on target, both schedule and cost wise. Let us examine the two major possibilities for the testing task on May 1: (a) the testing task is completed and (b) the testing task is not completed. The amount of actual effort expended is also a variable. This is shown in Figure 2, where the completion date of test task is marked as XXX and the AC for test is marked as NN.

Figure 2: Project Status on May 1

Major Task	Estimated Effort (BC) In person hrs.	Actual Effort Expended (AC) In person hrs.	Estimated Start date	Actual Start date	Estimated Completion date	Actual Completion date
Requirement	100	105	Jan 10	Jan 10	Feb 15	Feb 15
Design	76	65	Feb 10	Feb 10	March 10	March 5
Implement	150	150	March 1	March 1	April 15	April 15
Test	90	NN	Feb 10	Feb 15	May 1	XXX
Integrate	8	0	May 2	-----	May 5	-----

Consider the first case where the testing task is completed on May 1. Then there are three further sub-divisions: (i) the actual cost is NN= 90 person hours as planned, (ii) the actual cost is NN > 90 person hours, (iii) the actual cost is NN < 90 person hours. The best possibility is case (i) that NN = 90 as planned, in which case the SPI = 416/416 = 1 and CPI = 416/410 >1. The 5 days delay in test starting day caused no harm because the project is on schedule with a slight cost under-run. In case (ii) let us assume that we expended 100 person hours. Thus NN would be 100 and SPI = 416/416 = 1 and CPI = 416/420 <1. The delay caused a slight cost over-run, perhaps to make up for the schedule delay. For situation (iii), consider NN to be 80. Then SPI is still 1, but CPI = 416/400 > 1. The delay caused no harm in case (iii). Perhaps, the estimated BC for testing activity was a bit high to start with. The three scenarios demonstrated here show that in two of the three cases, the five day delay in starting the test task did not adversely affect the project.

Now consider the second case where the testing task is not complete on May 1 as planned. While we can not tell when the task will be complete and what the actual final effort would be, we can still look at the situation on May 1. We would have the same three

sub-divisions of actual effort expended on May 1. First consider (i) where NN = 90 units expended. Now follow the strict EVM metric rule. Then BCWP is really 326 because test task is not complete on May 1, and BCWS is 416. Thus SPI is 326/416 < 1 and CPI = 326/410 <1. This says that the project is behind schedule with a cost over-run and potentially even higher cost over-run. This is bad, but it is the correct status on May 1. For situation (ii), again, assume that NN = 100 units expended even though the task is still incomplete. Then SPI = 326/416 <1 and CPI = 326/420 < 1. This says that the project is behind schedule but has an even higher cost over-run than case (i). This is also a negative status, but a correct one. Lastly, consider case (iii) where NN = 80. Again, SPI is the same less than 1, and CPI = 326/400 < 1. This project status report is pretty much the same as case (i) and (ii); that is, the project is behind schedule and over-run in cost.

When we fast forwarded from March 15 project status day to May 1, we saw that of the six possible scenarios four resulted in negative outcomes. For the three scenarios associated with the test task missing the May 1 completion date, the project was both behind schedule and over-budget. EVM, being a technique that is based mostly on effort at completion, does not readily provide a view into the potential problems of missing the task start date. The earlier a potential problem is squashed, the less likely will the problem mushroom into some uncontainable situation as shown in this example of test task missing the February 10th start date. Therefore, in using EVM, project managers can not just depend only on numeric figures such as SPI and CPI. One must still track not only the end dates, but also the actual start dates of the tasks and perform some forward projections when a task start date is missed.

4 Concluding Remarks

In this paper, we examined EVM as a project management technique for software development. In particular, we focused on its capability in portraying the current status and in projecting the future. As expected, future prognostication is a difficult task for most management techniques. We focused on two levels, macro and micro levels, of looking beyond just current status. Through this exploration we have proposed several improvements. One improvement at the macro level is a better in-project forecasting mechanism for Estimate at Completion of Cost (EACc) which allows one to vary the Cost Performance Index (CPI) depending on what and the nature of the remaining tasks. The other, at the micro level, is to look beyond just the SPI and CPI indices of EVM. Project managers must be mindful of the actual start and completion dates of tasks.

5 References

[1] W.F. Abba, "Earned Value Management-Reconciling Government and commercial Practices," Project Management, Special Issue, January/February 1997, pp 58-67.

[2] P.G. Armour, "The Business of Software - How We Build Things," Communications of ACM, January 2013, vol.56, No1, pp 32-33.

[3] I. Attarzadeh and O.S. Hock, "Implementation and Evaluation of Earned Value Index to Achieve an Accurate Project Time and Cost Estimation and Improve Earned Value Management System," International Conference on Information Management and Engineering, Kuala Lumpur, Malaysia, April, 2009.

[4] D. Christensen, "The Estimate At Completion Problem: A Review of Three Studies," Project Management Journal 24, March 1993, pp 37-42.

[5] H. Erdogums, "Tracking Progress through Earned Value," IEEE Software, September/October 2010, pp 2-7.

[6] Office of Secretary of Defense, Earned Value Management, www.acq.osd.mil/evm, accessed December, 2012.

[7] Q.W. Fleming and J.M. Koppelman, Earned Value Project Management, 4th Edition, Project Management Institute, Inc., 2010.

[8] T. Sulaiman, B. Barton, T. Blackburn, "AgileEVM – Earned Value Management in Scrum Projects," Proceedings of the AGILE 2006 Conference, Minneapolis, USA, July 2006, pp 7-16.

[9] F. Tsui, Managing Systems and IT Projects, Jones and Bartlett Learning, 2011.