Earned Value Management approach in the Software Project Management

Evgenii Morozov

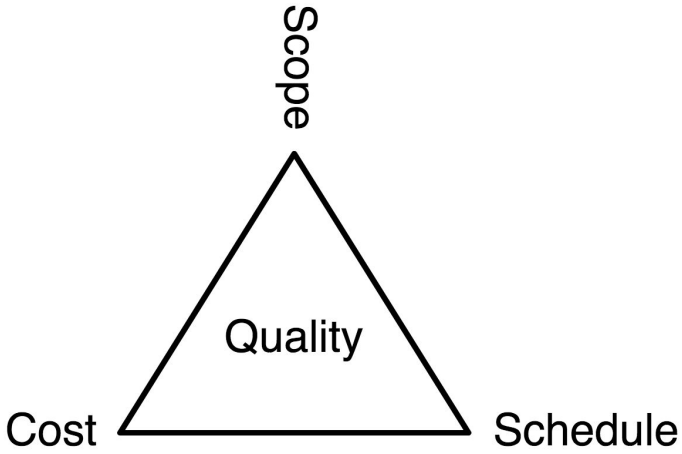
Earned Value Management (EVM)

Earned Value Management is a methodology used to measure and communicate the real physical progress of a project and to integrate the three critical elements of project management:

– scope

– time

– cost management



Purposes of an EVM

The purpose of an EVM system is to provide answers to project managers on questions such as:

- What is the difference between **budgeted** and **actual costs**?
- What is the **current** project status? Ahead of schedule or schedule delay?
- Given the current project **performance**, what is the expected remaining time and cost of the project?

The Challenges

Challenge – Innovation and Prototype

– Many software projects may be exploring functionality that has never been developed before or has never been applied to a particular functional area.

– Prototypes and innovations are not always successful during the execution of a software project – the result is often unplanned rework

– The calculations for the budget at completion (BAC) of the project as well as estimates for completion dates become unstable, resulting in non-sense answers in many EVM systems.

Challenge – Defect Discovery and Resolution

– All software has defects (bugs). During the process of development the discovered defects must be resolved or fixed.

– The nature and size of the defects discovered can be hard to predict, resulting in a large amount of volatility and long delays with no positive progress

– This then results in performance indices that trend downward and without reaching the goal. This reduces the ability of a software project to accurately forecast.

Challenge – Architectural Changes

– When the development of a software begins: certain architectural choices are made. As time and team experience progresses some of the architectural choices may need to be changed.

– An architectural change can result in unplanned rework. This unplanned rework can result in no project progress or even negative project progress for an extended period of time.

– To compound the problem further, the architectural changes often result in a large number new defects being introduced into the software

The Solutions

Solution – Measuring Volatility through Metrics

– In order to ensure that the values of the metrics collected and communicated accurately reflect the project and it's progress, the volatility of the project and the measures taken must be examined.

– Examining not only the traditional earned value management metrics, but also providing additional measures to gauge the volatility or stability of the project processes is a technique that is addressed by the PBEVM approach.

Solution – Task Definitions

– On software projects a key to being able to accurately track work progress with earned value is a set of guidelines used for defining task duration. On projects that are small or large, if the duration of an individual task is too long, the tracking of that task becomes highly inaccurate.

– Confidence Intervals: Planning for tasks should never be done using the nominal values. A common confidence interval for software projects is on the order if 80%.

– Short Duration Tasks: By keeping the duration of the tasks that are scheduled to a short duration, no more than two to four weeks, it ensures that schedule estimation errors are minimized and localized.

Solution – Scheduling Techniques

– Resource Loading: every task must be assigned to a resource in order to ensure that the task list can be measured against the available resources.

– Weighted Milestones: One effective set of weighted milestones is to grant 25% for starting a task, 50% once significant progress has been made that demonstrates an understanding of the challenges involved in the task, 75% on the first report of completion, and then 100% upon confirmation of completion of the task.

– Task Queuing Technique: the technique of creating a set of collectively exhaustive tasks that are mutually exclusive based upon the specific resource that has been assigned to them.

Solution – Incremental Implementation Approach

– High Risk Tasks: By taking those high risk tasks and moving them up into the schedule as early as possible, if the risk does occur, then there is more flexibility in forming a response.

– High Value Tasks: It is important to schedule the functionality that is of the highest value to the customer early in the project. In this way, each incremental delivery has value to the customer.

Applying The Solutions

Applying – Innovation and Prototypes

– Effective Tracking of Innovations and Prototypes: The approach here should be to focus the scheduling of the task on the various phases of the process of discovery that can be estimated.

– Reducing the Impact of Innovations and Prototypes: Using the incremental approach to bring high risk items into the development process early helps alleviate the amount of rework that may occur later in the project.

Applying – Defect Discovery and Resolution

– Development Defects: These defects are discovered and resolved during the development process so their resolution should be tracked as a part of the work package that describes the work associated with a particular release.

– System Test Defects: In an integrated development environment where there are dependencies on the work of different development staff, there is a need to track these types of defects in the repair cost account.

– Reducing the Effect of Defects Discovered: If there are a large number of defects that are discovered and need to be resolved, the resources to correct those defects are usually the same set of resources that are used for the generation of new features in the software system.

Applying – Architectural Changes

– Effective Tracking of Architectural Changes: In order to effectively track the progress of architectural changes, the adaptation costs for the architectural change to the existing modules should be captured under the development cost account and as part of the specific deliverable that the architectural change is needed to enable.

– Reducing the Effect of Architectural Changes: In order to reduce the effect of architectural changes the iterations should be sized in order to give the team time to react to future instances of architectural changes in future releases. The goal should be that when there is a large amount of instability in the architecture, the releases that are planned and managed should be shorter in duration.

Conclusion

There are a variety of challenges that face software projects when they are working to deliver high quality software on time and within budget constraints.

Many of those challenges are related to a large degree of uncertainty, either in schedule duration, quality factors, or design issues.

By applying techniques that help quantify the nature of the uncertainty, separate the distribution of uncertainty in project schedule and insulate the project budget from the effects of the uncertainty, projects can be successful in gaining value from EVM.