# Using Earned Value
# Part 1: Planning Software Projects


## Abstract

We've all experienced it too often. The first 90% of the project takes 90% of the time, and the last 10% of the project takes 90% of the time again. How can 90% of our projects be 90% complete 90% of the time? This happens because we use planning and tracking methods that allow far too much subjectivity.

Earned Value is a project planning and tracking method that removes much of the subjectivity from the process.

- In this first article (of two), we will discuss the principles behind Earned Value planning and tracking, and learn how to create an Earned Value plan for a software project.
- In the second article, we will learn how to track status against that plan, and how to handle problems like accounting for unplanned tasks.


## Principles Behind Earned Value Planning and Tracking

The Earned Value system removes much of the subjectivity from our project tracking process by measuring project status in terms of discrete tasks. Each task is either complete or not, and each task earns a set value when it is complete. By breaking the work down into small enough tasks, we can avoid the problem of estimating percent complete by asserting that each individual task earns no value until it is complete.

The objectivity that this method brings to our project tracking more than compensates for its slight conservative bias (showing no progress on tasks that are partially complete). Of course, given our normal tendency to underestimate our work, a conservative bias in project tracking can hardly be called a problem.

The Earned Value system is based on a set of principles that govern how the Earned Value plan is constructed and how the project earns value against that plan.

### *Plan to an appropriate level of detail*

This is perhaps the most critical issue in planning. The further into the future you attempt to plan, the less accurate your plan will be because of the amount of uncertainty involved. The more uncertainty there is, the less detail will be useful in the plan. So we should avoid wasting effort planning to an unreasonable level of detail.

Each situation will require your judgement, but the following guidelines may be helpful in getting started. (The examples in these guidelines are of a project that is beginning the implementation phase of release one. Development of all of the functionality for release one will be done in four iterative steps, and will take a whole year. A second release of additional functionality is also being planned for the following year.)

 

- When planning more than 12 months into the future, plan in terms of major milestones and aggregated tasks.

  In our example project, our plans for the second release should be in terms of broad feature sets and target acceptance and release dates, but should not be more detailed than that (until the time gets closer).

- When planning 3-12 months into the future, you should plan all milestones and tasks. The granularity of the tasks need not be small; tasks as large as 5-10 weeks would be quite useful.

  In our example project, plans for the second through fourth development iterations should be in terms of components or modules, and should identify all important milestones, like baseline dates and customer reviews.

- When planning less than 3 months into the future, you should plan all milestones and tasks down to a very detailed level. Each task should be small enough to complete within one week.

  In our example project, the first iteration of development work should not only be broken down into its discrete elements (Class A, Frame X, Datafile W), but it should include all of the necessary activities for each of them (e.g. detailed design, peer design review, write the code, peer code review, unit test, etc.)

Earned Value is for detailed planning, as described in the last bullet above. This is because the point behind Earned Value is to produce a plan that we can use on a week-to-week basis to understand our current status. There is no tracking to be done against future plans. We can only track against our current planning window, so that is where Earned Value is useful.

The remainder of these two articles will deal *only* with detailed planning and tracking in the relatively short-term window of our current work.

### *Available effort hours*

When building an Earned Value plan, we must accurately determine how much effort is available to put against the tasks. No one spends the entire workday on productive work. Besides the continuous parade of meetings, we have interruptions from colleagues, bugs to fix in prior products, breaks to clear our heads, time lost to context-switching, and idle chatter over the cube wall. PSP-trained engineers who carefully track their time and interruptions generally find that in a 40-hour work week, they must work diligently to exceed 15 hours of productive time on task.

In order to build a realistic plan, your planned available effort must be net of:
- Holidays, vacations and other time off,
- Effort already promised to other projects or activities,
- Time for inevitable maintenance or support of prior products, and
- Normal daily overhead and wasted time.

 

The first time you attempt to do this for a project, you will likely over-estimate your available task hours, so be conservative. After that, you can use your actual history on prior projects to make more accurate estimates of your available hours.

### *Appropriate task detail*

As we mentioned above, tasks should be quite small, compared to the level of planning most of us are used to. When choosing tasks, or breaking larger tasks down into smaller ones, we should consider these things:

- Each task should represent no more than about a week of available hours (as discussed in "Available effort hours", above). Since we will be evaluating our status based on the tasks completed each week, a single task that takes multiple weeks to complete will result in unknown status during the interim weeks. In order to earn value each week, we must complete one or more tasks each week.

- Each task must have clear completion criteria. There must be no question as to whether the value for a task has been earned or not.

- Each task is assigned a value that is based on the effort expected to complete it. This gives an appropriate weight to each task so that many small tasks might have the same value as a single larger one.

### *Earning Value*

The heart of the Earned Value system is in how the value is earned. The principles behind earning value are:

- The value of a task is earned when the task is 100% complete. No partial credit accrues for partially completed tasks.

  This rule eliminates the subjective practice of estimating "percent complete". When tasks are small enough, there is no need to think in terms of partial credit; the task is either complete or it is not.

- When a task is complete, you earn the *planned* value, even if the actual effort was different from plan.

  Recall that the planned value is based on the estimated effort for the task. Since the entire plan and schedule is built based on the estimated effort for all of the tasks, the status against the plan must be measured in the same terms. If your estimates are consistently high or low, then the accumulating error will become evident (as we will discuss in the second article in "Understanding Status").

- You earn no value for unplanned tasks.

 

It is inevitable that you will overlook some tasks when creating your earned value plan. The only way to understand our status against our plan is to avoid polluting the actual data with information that is foreign to the plan. We will discuss this issue further in "Accounting for unplanned tasks", in the second article.

These principles provide a sound basis for creating a useful plan and objectively measuring status against it.

## Creating an Earned Value Plan

The steps involved in creating an Earned Value plan are not materially different from any other planning process. (Identify the tasks to be done, determine the available effort, and use that information to compute a usable plan.) The critical difference is in how those steps are done and the guidelines that are applied. (Please refer to the example Earned Value plan in Appendix A to understand the descriptions that follow.)

### *The Work Breakdown Structure*

The first step in the process is to list all of the tasks that you must do. These tasks should be listed in a very fine level of detail. That is, instead of simply listing "Develop Module A", you might list these things for Module A: "Detailed Design", "Detailed Design Inspection", "Finalize Detailed Design", "Code", "Code inspection", "Finalize Code", "Compile", "Unit Test".

While this will seem like a lot of different tasks, it is the level of detail that you need, both to make accurate plans and to earn value every week. You should refer to any available process descriptions and historical data to identify all of the tasks that are involved in producing each thing you will produce.

You should list your tasks in the order in which you expect to complete them. While you will not be likely to exactly follow the order you plan, this will help you to understand your status against the plan (as we will discuss in the second article).

### *Effort Estimates*

You next must estimate the effort that each task will require. Rather than attempting to do this estimate directly, you should first estimate the size of the thing you will produce, then estimate the effort based on that. While this seems like the hard way to make an estimate, it provides several benefits:

- You will produce better effort estimates. This is because explicitly considering the size of something will help you to better gage the effort it will require.
- You can make better use of historical data. After estimating the size of what you will produce, you can find something similar that you produced in the past, and use your past experience to guide your estimate.
- You can improve your estimating accurace more easily. When your effort estimate is wrong, you can check to see if the thing was larger than you expected, of if you simply couldn't do the work as quickly as you expected.

 

After estimating the size of each thing you will create, estimate the effort each task will require, based on that size. If you expect Module A to be 200 lines of code (LOC), then you might expect to spend 2 hours designing it, an hour reviewing the design, and so on.

The final step in estimating the effort is to check for tasks that are too big. As we said, each task should be small enough to complete within a week of available hours. So, if you expect to spend 18 hours per week on project tasks, each task should require fewer than 18 hours. Any task that is larger than that should be broken down into sub-tasks in your work breakdown structure. For example, if you expect unit testing of Module A to take 25 hours, you should partition your set of unit tests into two independent groups, then estimate and track each of them as separate tasks.

When you are done, add up the total hours for all of the tasks. This is your planned effort for all of the listed tasks.

### *Effort Available*

Now that you have identified the effort that will be required to complete all of your tasks, you must determine when that effort will be available. You do this by using your calendar.

Beginning with the first week you will work on the tasks you are planning, step through your calendar and estimate the number of productive task hours you will spend on the project each week. Record your estimated hours for each week, and keep a cumulative total. Don't plan for any project time on holidays, vacations, or days of special activities (like training or company meetings). Continue walking through your calendar until your cumulative effort available exceeds the total effort for your planned tasks.

At this point, you have your best possible completion date. Because this method does not take into consideration dependencies among tasks, or the effects of other people's work, your actual completion date may be later than this snapshot indicates. But you can be certain that you can not complete all of the tasks you listed in <u>less</u> time without some form of heroic effort. (And you should never <u>plan</u> on heroic effort!)

### *Compute the Earned Value Plan*

(Please refer to Appendix A for an example of an Earned Value plan.)

You now have the information you need to compute your earned value plan:

- Task Effort = $TE_m$ = The estimated amount of effort for task m.
- Total Effort = TotE = sum($TE_1$, $TE_2$, …, $TE_{last}$)
- Week Effort = $WE_n$ = The estimated effort available for week n.
- Cumulative Effort for week n = $CumE_n$ = $CumE_{n-1}$ + $WE_n$

The mechanics are as follows:

- For each task in your work breakdown:
  - Compute and record the planned value:
    $PV_m = TE_m / TotE$

 

- Compute and record the Cumulative Task Effort:
  $CTE_m = CTE_{m-1} + TE_m$
- Record the expected completion week:
  $CW_m$ = the first week where $CumE_n > CTE_m$

- For each week, record the expected value earned:
  $ExV_n$ = sum(PV) for all tasks where $CW_m$ is on or before week n.

With all of the planning values complete, you can now create graphs that will depict how you expect your project to progress. The two most useful graphs are shown in the example in Appendix A. They are:

- Planned Effort by week – shows the cumulative effort growing week-by-week for as many weeks as are planned.
- Planned Value by week – shows the cumulative planned value growing week-by-week for as many weeks as the tasks require.

These two charts are the most interesting because they are the most useful for tracking the project, as we will see in the next article.

## Conclusion

Earned Value planning and tracking is a valuable method for improving our ability to accurately plan and track our projects.

- By providing a set of specific guidelines and set methods, it allows even the novice user to produce more accurate plans than before.
- By eliminateing subjectivity, it provides a reliable method for understanding project status.
- By maintaining a record of both your plan and your actual performance, it allows you to understand your planning errors and improve your ability to plan future projects.
- By recording actual data, it provides a basis for making more accurate estimates the next time.

Many individuals and organizations have adopted Earned Value and realized significant benefits. (The Personal Software Process teaches programmers how to use Earned Value to plan and track their personal work. And the Team Software Process makes effective use of Earned Value to help teams to achieved order-of-magnitude improvements in their project planning accuracy.) If your project plans are not as useful as you would like, then Earned Value should be of interest to you.

 

## Appendix A – Example Earned Value Plan

### *Task List*

| | | Estimated | | | | | Actual | | |
|---|---|---|---|---|---|---|---|---|---|
| WBS# | Task Description | Size | Effort Hrs | Cum Effort | Plan Value | Week | Effort | Size | Week |
| **1** | **Module A** | **1000 LOC** | | | | | | **874** | |
| **1.1** | **Detailed Design** | | | | | | | | |
| 1.1.1 | Write Detialed Design | | 10 | 10 | 8.4 | 1-Nov | 8.25 | | 1-Nov |
| 1.1.2 | Detailed Design Review | | 5 | 15 | 4.2 | 1-Nov | 4.75 | | 1-Nov |
| 1.1.3 | Finalize Detailed Design | | 2.5 | 17.5 | 2.1 | 8-Nov | 1 | | 1-Nov |
| **1.2** | **Code** | | | | | | | | |
| 1.2.1 | Write Code | | 10 | 27.5 | 8.4 | 8-Nov | 12 | | 8-Nov |
| 1.2.2 | Code Review | | 5 | 32.5 | 4.2 | 15-Nov | 5.25 | | 15-Nov |
| 1.2.3 | Finalize Code | | 2.5 | 35 | 2.1 | 15-Nov | 3.5 | | 15-Nov |
| 1.2.4 | Compile | | 2.5 | 37.5 | 2.1 | 15-Nov | 3.5 | | 22-Nov |
| **1.3** | **Unit Test** | **50 cases** | | | | | | **52 Cases** | |
| 1.3.1 | Write Test Cases | | 12.5 | 50 | 10.5 | 22-Nov | 20.5 | | 6-Dec |
| 1.3.2 | Run Test Cases | | 12.5 | 62.5 | 10.5 | 6-Dec | 10 | | |
| **2** | **Module B** | **700 LOC** | | | | | | | |
| **2.1** | **Detailed Design** | | | | | | | | |
| 2.1.1 | Write Detialed Design | | 7 | 69.5 | 5.9 | 6-Dec | | | |
| 2.1.2 | Detailed Design Review | | 3.5 | 73 | 2.9 | 6-Dec | | | |
| 2.1.3 | Finalize Detailed Design | | 1.75 | 74.75 | 1.5 | 6-Dec | | | |
| **2.2** | **Code** | | | | | | | | |
| 2.2.1 | Write Code | | 7 | 81.75 | 5.9 | 13-Dec | | | |
| 2.2.2 | Code Review | | 3.5 | 85.25 | 2.9 | 13-Dec | | | |
| 2.2.3 | Finalize Code | | 1.75 | 87 | 1.5 | 13-Dec | | | |
| 2.2.4 | Compile | | 1.75 | 88.75 | 1.5 | 20-Dec | | | |
| **2.3** | **Unit Test** | **60 cases** | | | | | | | |
| 2.3.1 | Write Test Cases | | 15 | 103.75 | 12.6 | 10-Jan | | | |
| 2.3.2 | Run Test Cases | | 15 | 118.75 | 12.6 | 17-Jan | | | |

### *Available Effort*

| | Expected | | | Actual | | | Projected | Value |
|---|---|---|---|---|---|---|---|---|
| Week of | Avail Hrs | Cum Hours | Planned Value | Effort Hrs | Cum Hours | Earned Value | Value | Per Wk |
| 1-Nov | 15 | 15 | 12.6 | 14 | 14 | 14.7 | | 7.0 |
| 8-Nov | 15 | 30 | 23.1 | 12.5 | 26.5 | 23.1 | | |
| 15-Nov | 10 | 40 | 31.5 | 10 | 36.5 | 29.4 | | |
| 22-Nov | 15 | 55 | 42.0 | 13 | 49.5 | 31.5 | | |
| 29-Nov | 7 | 62 | 42.0 | 6 | 55.5 | 31.5 | | |
| 6-Dec | 15 | 77 | 62.9 | 13 | 68.5 | 42.0 | | |
| 13-Dec | 10 | 87 | 73.2 | | | | 49.0 | |
| 20-Dec | 15 | 102 | 74.7 | | | | 56.0 | |
| 27-Dec | 0 | 102 | 74.7 | | | | 63.0 | |
| 3-Jan | 0 | 102 | 74.7 | | | | 70.0 | |
| 10-Jan | 15 | 117 | 87.3 | | | | 77.0 | |
| 17-Jan | 10 | 127 | 100.0 | | | | 84.0 | |
| 24-Jan | | | 100.0 | | | | 91.0 | |
| 31-Jan | | | 100.0 | | | | 98.0 | |
| 7-Feb | | | 100.0 | | | | 100.0 | |

### *Planning Charts*