Earning Function Points in Software Projects

Robert J. Muller, Chief Information Officer
ValueStar, Inc.
1120A Ballena Blvd.
Alameda, CA 94501-3682
Phone: (510) 814-7191 ✉ E-mail bmuller@valuestar.com

Abstract

Measuring the progress of a software project requires a metric of the value the project creates. Progress implies a plan against which to measure. The concept of earned value provides a clear metric of project progress relative to a plan. For this purpose, the usual cost metrics for earned value are less interesting in a software project than other measures of value. In particular, function points provide a measure of value that earned value metrics can easily use to compare what you've created to what you plan to create.

Value

Software projects create value.

Surely, this statement is a truism. What development manager would take pride in a software project that did not create value? Nevertheless, the statement is not as straightforward and obvious as it might appear. To illustrate, ask yourself a question--the same question philosophers have been asking since there was philosophy:

What is value?

To measure value, in particular, you must have some clear definition of what value is. Without penning a philosophical treatise, take as a starting point Gerald Weinberg's definition of software quality [Weinberg 1992, p. 7]:

Quality is value to some person .

The question of value thus is relative to people. In the case of a software project, the people who define value are the project's stakeholders: customers, marketing, executives, and even the developers and managers of the project. Stakeholders inject their needs into the project through expectations that the project manager collects and balances during the planning phases of the project. Expectations translate into requirements, and requirements translate into use cases, functional specifications, or other analytical statements of the scope of the project. By the end of the initial planning stage, the project manager should have a clear statement of the scope of the project and its value relative to the project stakeholders' expectations [Muller 1998].

Given a plan, the project manager must then manage the project, controlling the delivery of the planned value to the stakeholders against a schedule and budget to a defined level of quality [Kerzner 1995, PMI 1996]. This paper develops metrics for measuring progress that the project manager can use in managing delivery of value.

It should be clear from this discussion that measuring the amount of code you produce is not relevant to controlling the project. Code is not value, what code does for stakeholders at what price is value. How do you measure that?

The approach this paper takes is to break the question into two parts. First, how do you measure cost and functionality? Second, how do you relate delivered functionality to the stakeholder's expectations?

Measuring cost is relatively well understood in the project management world. Measuring functionality is not, since many projects still measure output in terms of lines of code rather than function points or other direct measures of functionality.

Measuring delivered value is even less well understood, although the project management world defines a clear set of metrics for it: earned value [Kerzner 1995, PMI 1996]. Earned value is the value of the product a project delivers as part of its planned activities. This definition specifically measures value against expectations. Value not planned is not earned value. Earned value thus attacks two project management problems:

- Scope creep, the delivery of unplanned value
- Schedule and budget overruns, the delivery of planned value late or at too great a cost.

Baselining a Project

To control a project, you must first draw a line in the sand against which to measure progress. The planning process does this by defining various baselines.

A *baseline* is a particular set of versions of systems that satisfy a set of exit criteria [Muller 1998]. The systems are the cost, schedule, or product systems you build during the project. The exit criteria are the quality, schedule, or cost metrics you will use to determine when a subsystem is complete.

The cost-oriented baseline is the project budget. Each activity in the plan has a budgeted cost based on estimated effort, labor and overhead costs, and material costs. The budget also has a time component that relates budgeted costs to milestones in the project schedule. By a certain milestone, the project manager intends to have spent a certain amount of project resources.

The value-oriented baseline is similar. Each activity in the plan has a planned value based on analysis of how the activity contributes to satisfying stakeholder expectations. Again, milestones provide checkpoints at which the project will have delivered a specific amount of value.

Both of these baselines depend on a lower-level baseline of the software (and other types of) objects the project intends to deliver. At each milestone, the system contains a planned set of software systems, usually called the system configuration. The versions of software objects in this baseline (the alpha baseline, beta baseline, production baseline, and so on) provide a certain level of value to the stakeholders.

It's important to realize that baselines are themselves versioned systems. As the environment and understanding of the software system changes, the baselines must change. Tracking the versioning of baselines is a large part of the job of the project manager as the project progresses. It's also part of the project manager's job to ensure that the baseline changes in a controlled way.

Baseline budget and schedule are the basis for measuring earned value. The changes to baselines thus directly affect how you measure earned value and hence how you measure project progress.

Cost

The idea behind the standard project management concept of earned value is to combine schedule and cost in a measurable system of project control [PMI 1996, Kerzner 1995]. The original government standards that developed the approach named it Cost/Schedule Control System Criteria (C/SCSC) for that reason [Fleming 1983].

It's important to note, however, that the concept of earned value usually used by project managers does not directly address value at all, despite the name. Rather, the metric should be called earned cost, because what you're really measuring is the budgeted cost you've incurred, not the planned value you've earned.

Cost is the amount of money you spend to develop a software system. Cost accounting is a well-known, comprehensively taught subject. To compute the cost for an activity, for example, you add up the following components:

- Fixed Costs (rent, etc.)
- Variable Costs
  - Pay rate x hours worked x burden
  - Regular versus overtime
  - Bonuses
  - Unit cost x number of units

Function Points

A *function point* is a weighted count of the number of features in a software product [IFPUG 1994, Garmus 1996]. A key advantage of function points over other measures of software size is that you can count function points from the requirements specification (the complete one, not the informal one). Lines of code, the main competitor, requires the code be there before you can accurately count. You can use function points to predict effort, cost, and duration once you establish historical data for your organization.

Function point counting requires some training, the best source of which is the International Function Point Users Group and its conferences [IFPUG 1994]. Garmus and Herron have an excellent tutorial [Garmus 1996]. To count function points, you develop a complete functional specification that defines the scope of the system, then count these items for that system:

- **External Input** (for example, data entry forms)
- **External Output** (for example, reports)
- **External Query** (for example, find dialogs)
- **Internal Logical File** (for example, relational tables this application maintains)
- **External Logical File** (for example, other tables this application refers to but does not maintain)

Once you have counted an item, you weight it with a complexity factor based on the number of elements within the item (fields in data entry forms and reports, columns in tables, and so on). Summing the weighted counts gives you the raw function points. You then adjust that value with a weight for the environment which takes into account everything from data communications facilities to performance requirements to transaction rate to reusability to maintainability. The standard client/server database application system, for example, has a weight near one, while a real-time embedded system with special communications requirements may be significantly greater than one.

Value Baselines

The critical metric for earned value is planned, budgeted or baselined cost and value. Without a baseline for cost or function points, you cannot measure earned value, since you cannot measure planned/budgeted/baselined value. Again, various baselines provide such values for each activity in the project, with summations at milestones in the project schedule.

Cost is easy to allocate to activities. Each task gets an estimated cost based on its resources and outputs. But it is harder to estimate the value for an individual activity.

A key decision you must make is when to earn the value in a system. The easiest way to specify this is to earn value when you complete a subsystem that actually delivers the complete functionality expressed by the function points. For example, a subsystem in the functional specification contains a complete function that lets a user enter a specific transaction. Until that subsystem is complete, the user cannot enter that transaction, and so the system does not provide the specified value.

Since most software systems do not deliver value until the alpha release or later, this way of allocating value to tasks will not be of much help in controlling the middle part of a software project (the most important part). You can do two things about this.

First, deliver as much value as early as you can in the project. Instead of waiting to get a working system late in the project, get small parts of the system working as early as possible. These early milestones thus give you a way to earn value throughout the project as features become available. A good exit criterion for these milestones is whether the system is ready for a system test by quality assurance.

Second, you can allocate the function points for a subsystem among the activities using some kind of allocation algorithm. A straight-line method, for example, allocates the function points evenly between all the tasks that contribute to a subsystem. At a milestone, you then sum up all the function point fractions from completed tasks much as you would sum up the costs of the tasks. A weighted method might use the effort or cost estimate to weight the allocation of function points. You might also have some kind of technique for allocating function points based on your estimate of how much value the task actually contributes or how "complete" the task or subproject really is.

Note: I've had more success with straight-line methods than with weighted or "smart" methods of value allocation. Once you get subjective by introducing estimates, you are falling back on techniques that I've usually found feeble at best. You may have better estimating methods than I have seen, so by all means try it and let me know what you've done.

Earning Value

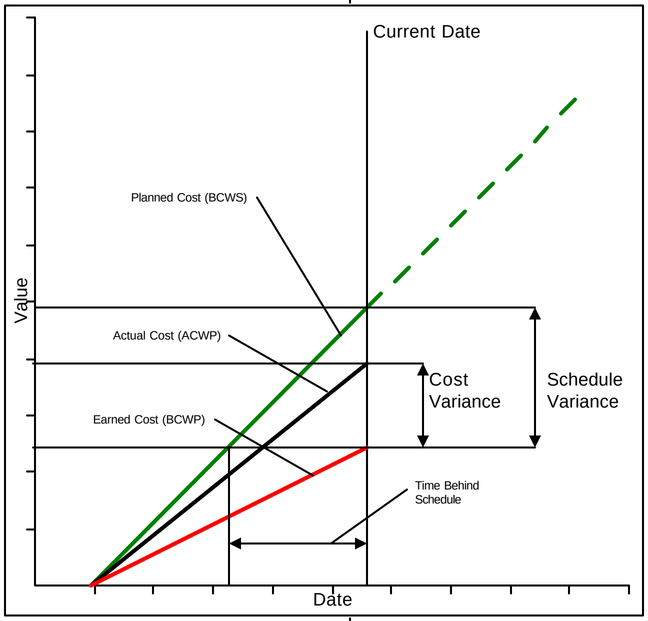
You've estimated costs. You've counted function points. You have a schedule with milestones. How do you actually measure the value you've earned? Value in function points or cost is only part of the story.

- **Time:** How does the timing of creating and delivering value affect it?
- **Cost:** How does the cost of creating and delivering value affect it?

There are two key principles to earned value.

- **You earn value when you deliver it.**
- **You only earn what you promised.**

First, whatever allocation method you use for apportioning value in function points or cost to the scheduled tasks of your project, you only earn value when you deliver something that has value. It is therefore a good idea to put milestones into your project schedule that represent actual, delivered value, then to measure earned value only at those milestones. It is important to realize that the milestone occurs when you deliver the value, not just because you've reached a date.



Earned value, on the other hand, uses the dates from the planned milestones, not the achieved ones. At any given date, earned value thus represents what you planned to achieve by that date, not what you've actually achieved. Whether you take measurements once a week or once a month, the earned value thus tells you how you're doing relative to what you planned to do. Changing the baseline, of course, affects the planned dates and thus changes the earned value.

Second, earned value is by definition *planned* value. If you didn't plan to deliver a function, it does not become a part of your earned value. This idea forces you to see how your system corresponds to stakeholder expectations and lets you control scope creep. **This is the fundamental idea behind earned value.**

Earned Cost Arithmetic

Now for some simple earned cost arithmetic. The standard metrics in the C/SCSC method measure various costs at a given milestone. Most project management software provides these metrics.

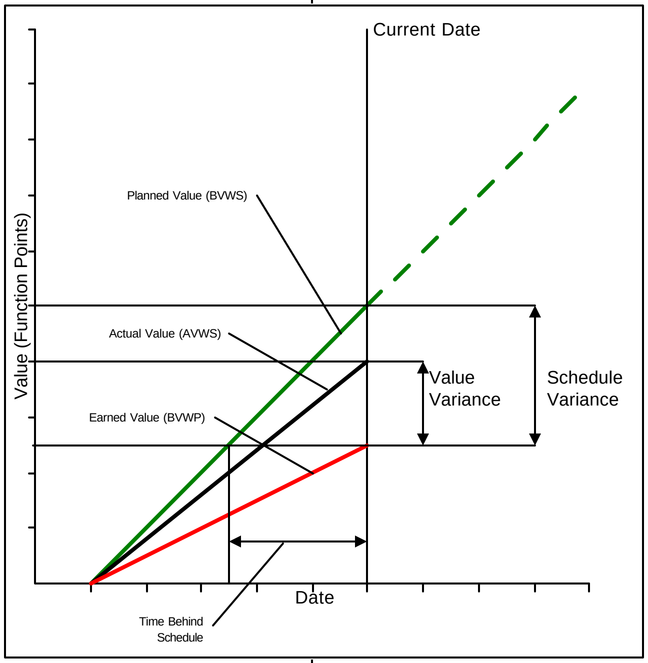
- Budgeted Cost for Work Scheduled (BCWS)
- Budgeted Cost for Work Performed (BCWP)
- Actual Cost for Work Performed (ACWP)

BCWS is the cost you've budgeted for work through the milestone date. For example, if the milestone takes place on November 3, and the budget baseline calls for spending $45,000, then $45,000 is your BCWS on Nov. 3.

BCWP is the originally budgeted cost of completed work at the milestone date. This is *earned cost.* That is, at the milestone date you've completed a certain number of the tasks you've planned. The planned cost in the budget baseline for those completed tasks is the BCWP. For example, given the planned $45,000 cost at the milestone on Nov. 3, say you've completed only 80 out of the 100 tasks you planned to complete. The BCWP is the amount of cost allocated to the 80 tasks, say $35,000. You've earned $35,000. Alternatively, you may have completed another 20 tasks ahead of schedule; then your earned cost might be $55,000, more than you planned.

Finally, ACWP is the actual cost of completed work. The other numbers count only the planned cost. This one measures the real cost you've incurred to reach the milestone. This number gives you the basis for some further derived metrics that compare actual to earned cost, such as cost variance and schedule variance.

Cost variance is BCWP - ACWP, the difference between planned work performed and actual work performed. This number tells you about money you've spent that you didn't intend to spend on the specific tasks you've completed. Schedule variance is BCWP - BCWS, the difference between planned cost of work performed and planned cost. This is the difference in value between what you planned to achieve and what you've earned at the milestone date. This number tells you about the money you haven't spent because you didn't do what you promised to do.



The Cost Performance Index is BCWP divided by ACWP, the ratio of earned cost to actual cost. This ratio tells you the cost efficiency of your project: how well you are using resources to get results. If earned cost is higher than actual cost, you are spending more than you planned to get the planned result. If earned cost is lower than actual cost, you are spending less than planned.

The Schedule Performance Index measures the time efficiency: BCWP divided by BCWS, the ratio of earned cost to planned cost. If earned cost is less than planned cost, it means you've spent less than you planned to on planned tasks, usually because you haven't done the tasks you've planned. If earned cost is higher than planned cost, you're ahead of schedule on task completion.

Value Arithmetic

Now for the equivalent earned value arithmetic. These metrics measure value in function points at a given milestone. You will usually need to track these on your own, as project management systems will not collect the metrics for you.

- Budgeted Value for Work Scheduled (BVWS)
- Budgeted Value for Work Performed (BVWP)
- Actual Value for Work Performed (AVWP)

BVWS is the value you've planned in function points for work through the milestone date. For example, if the milestone takes place on November 3, and the system baseline calls for having software that delivers 400 function points (FP) in value, then 400 FP is your BVWS on Nov. 3.

BVWP is the originally planned cost of completed work at the milestone date. This is *earned value.* That is, at the milestone date you've completed a certain number of the tasks you've planned. The planned value in the system baseline for those completed tasks is the BVWP. For example, given the planned 400 FP value at the milestone on Nov. 3, say you've completed only 80 out of the 100 tasks you planned to complete. The BVWP is the amount of cost allocated to the 80 tasks, say 350 FP. You've earned 350 FP. Alternatively, you may have completed another 20 tasks ahead of schedule; then your earned value might be 550 FP, more than you planned. If you are being conservative, of course, you might not earn anything at all in the case where you didn't complete the tasks that actually delivered the value.

Finally, AVWP is the actual value of completed work. The other numbers count only the planned value. This one measures the real value you've delivered at the milestone. It's a good idea not to assume that this value is precisely what the functional specification called for. Instead, go back and do another function point count on the delivered system. The new number may be radically different from the old one if you haven't been paying attention to the right things.

As with earned cost, this number gives you the basis for some further derived metrics that compare actual to earned value. Specifically, this metric tells you about value you delivered that was not planned. These are the things that developers thought might be nice to have, regardless of the features not being present in the functional specification. This tells you not just about unplanned scope creep but about uncontrolled software development.

The Value Variance is BVWP - AVWP, the difference between planned value of the work you've completed and the actual value of the work you've completed. This metric tells you about the difference between planned value and value achieved. Achieving less value than planned by a given date means that not only have you delivered less value than you wanted, you've actually delivered less value in each system. This is usually because you've dumbed down the specification without changing the baseline. Achieving more value than planned means that you overengineered the product. You use this to control your technical management of tasks.

The Schedule Variance is BVWP - BVWS, the difference between planned value of work performed and the original planned value of the work. This number tells you how far you are behind (or ahead) of the set of tasks you planned to achieve by a given date. It is a straightforward statement that you are behind or ahead of schedule. You use this to control your schedule.

The Value Performance Index (VPI) is BVWP divided by AVWP, the ratio of earned value to actual value. This ratio tells you the value efficiency of your project: how well you are containing the natural urges of engineers to give more value than desired. If earned value is higher than actual value, you are producing more value than you planned--overengineering. If earned value is lower than actual value, you are producing less value than planned--cutting features to make the schedule or the budget.

The Schedule Performance Index for value measures the same time efficiency as its cost cousin: BVWP divided by BVWS, the ratio of earned value to planned value. If earned value is less than planned value, it means you've produced less than you planned to on planned tasks, usually because you haven't done the tasks you've planned. If earned value is higher than planned value, you're ahead of schedule on task completion. Measuring the value-based SPI tells you about schedule efficiency in terms of value rather than cost. That is, SPI tells you how well you're keeping to schedule with respect to delivering value, as opposed to spending money.

Looking Ahead

Of course, just knowing where you are is not enough. You must then turn around and tell your stakeholders (and primarily your boss) where you're going. The earned cost and earned value indexes of efficiency can help quite a lot here.

Standard forecasting techniques use the basic assumption that, human nature being what it is, your efficiency in the past is going to be your efficiency in the future. The Budget at Completion (BAC) sums all cost baselined for the project. The Estimate at Completion as of the milestone date (EAC) is the BAC divided by the CPI at that date. The Variance at Completion (VAC) is BAC - EAC, the difference between the budget and the estimated cost given your current cost efficiency. If VAC is negative, the project is going to overrun its budget. If it's positive, the project is in good shape!

Moving to value from cost, Planned Value at Completion (PVAC) is the value you planned to deliver by the end of the project. Estimated Value at Completion is then the planned value divided by the VPI (value efficiency). The Value Variance is then the difference between planned value and estimated value. If this variance is negative, it means your project is going to deliver less value than you planned. This conclusion tells you to rebaseline to the lower value or to the higher costs or longer schedule required to increase the value. If the value variance is positive, it means your project is on track to deliver more than the stakeholders want, not necessarily a good thing in a competitive, time-to-market oriented world. Alternatively, a high value variance might indicate underspecification of requirements. Perhaps the engineers have uncovered hidden or mistaken requirements. This conclusion tells you either to exercise higher control over the development process to prevent excessive overengineering of the project deliverables or to revise the requirements specification and rebaseline the project.

Earning Points

Using earned cost lets you control your project with respect to schedule and cost. Using earned value lets you control your project with respect to the product you're delivering. Both have a place, and both are valuable contributors to project control.

Using function points lets you control value through a stable, well-understood measure. Instead of just approximating value with cost, earned value gives you a way to manage expectations and feature creep with real data.

References

[Fleming 1983] Quentin W. Fleming. *Put Earned Value (C/SCSC) Into Your Management Control System.* Humphreys & Associates, 1300 Quail St., Newport Beach, CA, 92660, 714-955-2981.

[Garmus 1996] David Garmus and David Herron. *Measuring the Software Process: A Practical Guide to Functional Measurements.* Yourdon Press, 1996.

[IFPUG 1994] International Function Point Users Group. *Counting Practices Manual, Release 4.0.* IFPUG, Blendonview Office Park, 5008-28 Pine Creek Drive, Westerville, OH 43081, 614-895-7130, (www.ifpug.org).

[Kertzner 1995] Harold Kerzner. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling. Fifth Edition.* Van Nostrand Reinhold, 1995. Pp. 813-829.

[Muller 1998] Robert J. Muller. *Productive Objects: An Applied Software Project Management Framework.* Morgan Kaufmann, 1998.

[PMI 1996] Project Management Institute. *A Guide to the Project Management Body of Knowledge.* Project Management Institute, 4 Campus Blvd., Newtown Square, PA, 19073 (www.pmi.org).

[Weinberg 1992] Gerald M. Weinberg. *Quality Software Management Volume 1: Systems Thinking.* Dorset House, 1992.

Robert J. Muller

Robert J. Muller is Chief Information Officer of ValueStar, Inc. His extensive project management experience ranges from client/server to object-oriented technologies. He is the author of the Morgan Kaufmann book Productive Objects, the forthcoming Database Design for Smarties, and the Oracle Press Developer/2000 Handbook. He is a member of IEEE, ACM, and the International Function Point Users Group.