Monitoring Scrum Projects with AgileEVM and Earned Business Value (EBV) Metrics

By Dan Rawsthorne, PhD, CST

Scrum is a popular project management framework for agile projects. Scrum projects are typically managed quite informally, with the only metrics used being various velocity metrics and burndown charts. Because these metrics only measure the speed of delivery, not the project's cost or the business value it generates, many project managers are resistant to Scrum. One of the major differences between traditional and agile projects is that traditional projects focus on delivering software that satisfies requirements, while agile projects focus on maximizing ROI through continuous feedback and re-planning. That is, the focus of agile projects is on business value rather than conformance to requirements, and so Earned Business Value (EBV) metrics can be crucial. Of course, it is also important to know how efficient and effective the team is in doing the work that provides Business Value, so Earned Value Management (EVM) metrics are also applicable. In fact, EVM metrics are easier to calculate and understand in agile environments than in traditional ones. In this paper, we describe how to use three management metrics—Cost Performance Index (CPI), Schedule Performance Index (SPI), and Earned Business Value (EBV)—to provide information to help manage an agile project. We also demonstrate (through a simulation) that (even) large changes in the SPI and CPI metrics don't necessarily mean a significant change in EBV.

INTRODUCTION

Scrum is a project management framework for agile projects. Scrum projects are typically managed quite informally, but there are some questions that managers seem to always want answers to, such as:

- Am I getting what I paid for?
- Am I getting it as fast as I expected?
- How much value am I getting?

We want to be able to answer these questions with confidence. In order to discuss metrics that address these questions, we first need to define some Scrum-specific terminology: sprint, feature, and story.

We would like to know how our project is doing all the time, but must settle for getting answers to these questions each *sprint* cycle, where a sprint is defined as a (fixed) time interval that begins with planning, contains work, and ends with review.

A *feature* is something about our project and/or product that we advertise or sell, and that we must do work on to provide. Typically, our product is described as a list of features, such as:

- Users Can Update Customer Information
- Supports the XXX Claims Form
- Added "Updated Customer Info" Reporting
- Improved User Interface
- Faster Rendering of Reports
- Improved Security and Protection against Hackers
- And so on…

Features are implemented in the product by working on stories, and the stories are said to *belong* to the feature.

A *story* is the fundamental unit of value, requirements, and work that is visible from both the inside (i.e., the team, product owner) and the outside (i.e., business owner, stakeholders, etc.) of the project. In other words, stories are used as the interface between the business and development sides of the project.

Stories are important since they are the smallest units of value that are managed. Stories have been described as "promises for conversation" by Alistair Cockburn [4], and I agree wholeheartedly. Basically, a story consists of four things:

1. **Description**: A short description of the goal to be achieved.
2. **Size**: A relative measure of difficulty of the story, measured in StoryPoints (SPs).
3. **Business Value (BV):** A measure of how much this story is worth to the business, which can either be relative or absolute. Not all stories have Business Value, but the ones that drive development do.
4. **Validation Criteria:** What it means to be "done" with this story. This is probably the most important attribute, as being "done" is what allows the team to earn the story’s SPs and BV.

AGILE EVM

In [2], Tamara Sulaiman *et al* pointed out that standard Earned Value Management (EVM) metrics could be used to help manage Scrum projects. Her realization was that the units of value for the purpose of EVM are *earned* StoryPoints (SPs); that is, the SPs of the Stories that have met their Validation Criteria.

In order for SPs to be an appropriate measure of value, they must represent something about the story, not about the team. In particular, they can’t explicitly represent a team’s effort, as this leads to the common trap of value being measured by hours delivered rather than product delivered. I recommend that SPs be a relative measure of innate difficulty of the story and that they focus on “how big,” not “how long,” the story is. Typically, I get this value by asking a question like: “How difficult is this story, given good code to work with and an ideal team?”

Once we have SPs to work with, we can calculate EVM metrics as we find in [2]. In what follows, I have modified the calculations from what is found in [2] in order to focus on the two metrics of most interest, CPI and SPI. However, the ones in this paper are the **same** AgileEVM metrics that we find in [2]. These are not new metrics; they are just calculated differently.

In addition, the calculations for these metrics are easily understood and involve information that should be readily available for a Scrum project.

In order to calculate AgileEVM metrics, we need to know a few things. First of all, we need to have expected, or baseline, values to compare our actuals to. We could have baselines that are complex—especially those that are based on team size/cost varying through time—but, in this paper, we assume consistency across sprints, as in the following table.

Table 1: Initial Baselines

Name	Definition
BV	Baseline Velocity – how many SP/Sprint we expect on average.
BC/SP	Baseline Cost per StoryPoint – how much we expect to pay for each SP on average.

Once we have these baseline values, we gather data for each sprint as we go. The following are the data we collect (actuals), and the derived metrics we use to compare to our expectations.

Table 2: Sprint Data Points

Name	Definition
n	Sprint number – starts at 1
PC_n	Points Completed – the SPs for the stories completed in the sprint
TPC	Total Points Completed – the total of all the SPs for all the stories completed to this point = $\sum^n_1 PC_i$
AV	Actual Velocity after n sprints = TPC/n
AC_n	Actual Cost for the sprint, typically given in Person-Days or Currency.
TAC	Total Actual Cost – the total of all the Actual Costs so far = $\sum^n_1 PC_i$
AC/SP	Actual Cost per StoryPoint – the average cost so far = TAC/TPC

With these definitions, we can now calculate our Earned Value Method (EVM) metrics. I don't want to go into all the gory details about how EVM metrics are calculated, so let's start off with the very basic definition—that of Earned Value. Basically, earned value is defined as "the value of completed work," according to the PMI. Since what we're using for completed work here is completed StoryPoints, earned value will be the amount that we should've spent for the number of StoryPoints that we've gotten done so far, based on our baseline.

That is, at the end of the n^{th} sprint, when we have completed TPC StoryPoints, our earned value is (TPC)(BC/SP), our total StoryPoints times our Baseline Cost per StoryPoint, the amount that we believe we should've spent for the StoryPoints we got. Another metric we'd like to have is Planned Value (PV), which is defined as how much we had planned to spend by now. After n sprints, we should have received nBV StoryPoints, at a cost of BC/SP apiece, so PV = nBV(BC/SP).

Once we have an EV, we can calculate all the standard EVM metrics. The two that we're really interested in are Cost Performance Index (CPI) and Schedule Performance Index (SPI), as we use these two metrics to calculate variances and revised estimates.

The standard definition of CPI is:

$$\begin{aligned} CPI &= EV/TAC \\ &= TPC(BC/SP)/TAC \\ &= (TPC/TAC)(BC/SP) \\ &= (BC/SP)/(TAC/TPC) \\ &= (BC/SP)/(AC/SP) \text{ or, in words,} \end{aligned}$$

Cost Performance Index (CPI) = (Baseline Cost per StoryPoint)/(Actual Cost per StoryPoint).

Likewise, the standard definition of SPI is:

SPI = EV/PV

= (TPC(BC/SP))/(nBV(BC/SP))

= (TPC/n)/BV

= AV/BV or, in words,

Schedule Performance Index (SPI) = (Actual Velocity)/(Baseline Velocity).

In other words, we have the following calculations.

Table 3: CPI and SPI Calculations

Name	Definition
CPI	Cost Performance Index—measures if we are getting the SPs we are paying for, CPI = (BC/SP)/(AC/SP)
SPI	Schedule Performance Index—measures if we are getting the SPs at the rate we expected, SPI = AV/BV

If the project went according to plan, both CPI and SPI would = 1 every sprint. If CPI > 1, we're spending less than we expected for each StoryPoint, if CPI < 1, we're spending more. If SPI > 1, then we're finishing earlier than expected, if SPI < 1, we're finishing later. These two metrics give us a fairly complete look at our StoryPoint production, and give us the ability to determine if we're "on track" or not.

The CPI and SPI metrics can be used in all the standard ways for our agile project: to determine new expected release dates, to determine efficiencies of our team, and so on. For example, if we were planning a 10-sprint release, and, after sprint 5, our SPI was .75, we would expect that our actual release would be after 10/.75 = 13.33 ≈ 13 sprints. Likewise, if our total budget was 5000 person-days and our CPI = 1.2, we would expect to actually spend only 5000/1.2 ≈ 4166 person-days.

Also note that SPI and CPI are useful even for maintenance-type projects with no fixed release date. We can incrementally calculate SPI and CPI on each sprint boundary or even as each story is completed. By looking at these two metrics, we can determine if our team is working as fast and efficiently as we planned them to do.

This is good stuff, but assumes that the purpose of the team is to deliver completed StoryPoints while, in actuality, its purpose is to deliver Business Value. Unfortunately, these metrics do not tell us anything about actual Business Value, they only tell us about how well we are getting our StoryPoints. We are answering:

- Are we getting StoryPoints at the rate we expected (SPI)?; and
- Are we paying what we expected to pay for them (CPI)?

This is the primary difference between traditional projects and agile ones. In traditional projects, we are expecting to deliver everything in our requirements document, so all we need to measure is efficiency and effectiveness of delivery. In other words, EVM metrics are enough for traditional projects. However, in agile projects, we don't know exactly what we're going to deliver. We're constantly evaluating to see if we have *enough*, if we've generated sufficient ROI, etc., so we need some way of determining the Business Value of what we've got so far in order to make that determination.

EARNED BUSINESS VALUE DEFINITIONS

In order to find out if we are getting the Business Value we expect/need, we have to use the completed stories' Business Values, not sizes. We define the EBV, at a given time, as "the percentage of the product's (release's) known Business Value that is currently earned[1]." In other words, we have the definitions in the following table.

Table 4: Business Value Metrics

Name	Definition
BVI(Story)	The Business Value Index of a story—the percentage of the total Business Value that the story "owns" = $\frac{BV(Story)}{\sum BV(Story)}$ where the sum is over all stories in the product / release, as appropriate.
EBV_n	Earned Business Value—the total of all the Business Value Indices for the completed stories = ∑ BVI(story) for all the stories completed in the first n sprints (it's a running total).

This last metric, EBV_n, is what we graph and call the Earned Business Value (EBV) graph. In [3], I have previously published a way to calculate BV(story) based on a Work Breakdown Structure that organizes the stories, and I omit that discussion here.

Now that we have these three metrics (CPI, SPI, and EBV) to track, let's look at a simulation. We will do this in stages:

1. Give the nominal (expected) curve for delivering Business Value for a feature.
2. Propose an "ideal" strategy for delivering a release, defined by a few features.
3. Show what might happen if the team does not deliver as expected, presenting CPI, SPI, and EBV graphs.
4. Note some of the issues and conclusions we can draw from this simulation.

DELIVERING BUSINESS VALUE FOR A FEATURE

When we earn Business Value for a feature, we are earning the BV for stories that belong to the feature. Typically, we think of earning the feature's Business Value (the total BV of the completed stories in the feature) as looking like an S-Curve [1].

What this S-Curve indicates is that stories that belong to the feature are delivered with the following strategy:

1. Deliver *architecturally significant* or *backbone* stories first, which have fairly small Business Values—but others depend on them.
2. Deliver *high Business Value* stories next, using a basically greedy algorithm.
3. Eventually deliver stories that have low Business Value, experiencing the law of *diminishing returns*.
4. It is vital to realize that the *critical mass*, or *minimal releasable*, version of the feature happens well short of the end, usually at a Business Value of .9 or so, as we transition from high Business Value to Diminishing Return stories.

[1] So it more correctly could be called the Earned Business Value Index (EBVI) ☺

Here is a typical S-Curve, as calculated by formulas found in [1].

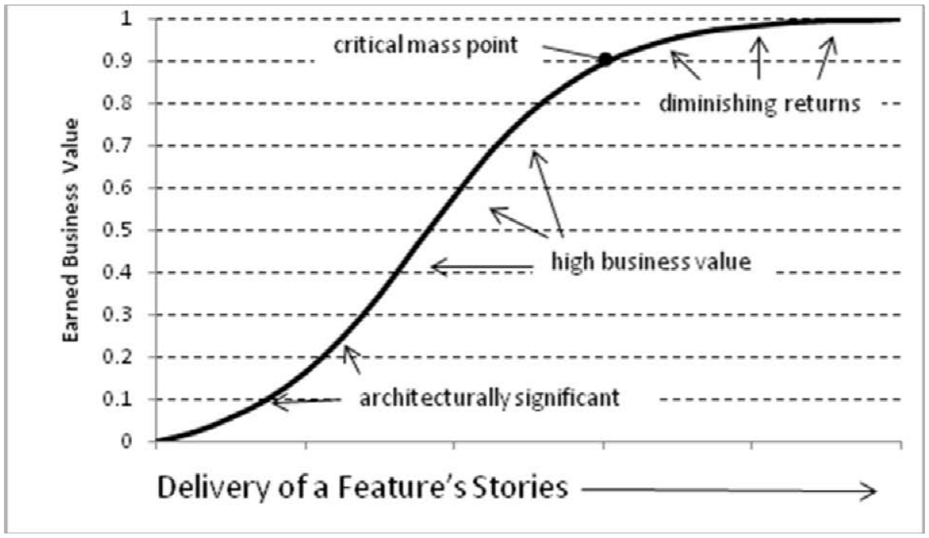


Figure 1: Typical S-Curve for Delivering Business Value for a Feature

The simulation we present uses this particular curve to determine BVI(story) for all the (feature) stories involved in the simulation.

Most of us have heard of the Pareto axiom, which is that we get 80% of the Business Value for 20% of the effort. I note here that the final 2/3 of this graph (after the architecturally significant stories) has the same basic shape as a 60/20 curve, which is less aggressive than an 80/20 curve and allows us less than perfect analysis and delivery. In other words, the 80/20 curve is what we notice in hindsight, looking back to see what *actually delivered* Business Value. The 60/20 curve is the best we can actually do when we move forward, as we "chew up" time and effort finding the stories that will provide Business Value.

SIMPLE RELEASE STRATEGY EXAMPLE

Assume we plan to deliver a product in 10 2-week sprints, with a team of 6. Assume that our baseline assumptions are that this team's velocity will be 50 SPs/sprint, and that we will get 50 days of effort from our team each (2-week) sprint (~8 days/person).

So, we have:

- BV (baseline velocity) = 50 SP/sprint
- Total SPs planned for = (50 SP/sprint)(10 sprints) = 500 SPs
- Total person-days planned for = (50 pd/sprint)(10 sprints) = 500 pd
- BC/SP (baseline cost per SP) = (500pd)/(500SPs) = 1 pd/SP

We have a total of 500 SPs to play with in this release and we budget them as follows to the three features we plan to deliver. We also show the BV for each feature in this table.

Table 5: Basic Release Strategy

Feature	Business Value (%)	Budgeted SPs
Feature 1	0.5	100
Feature 2	0.3	160
Feature 3	0.2	80
Chores	0.0	160
Totals	**1.0**	**500**

In this simple formulation, the feature labeled "chores" represents all the work we have to do that doesn't produce *any* Business Value, such as team infrastructure and training, analysis, and so on. Chores take time and effort, but the value they produce is not Business Value and is usually budgeted as 1/3 (or so) of the total budget, as we do here. Note that we *do not* have a baseline effort for any of the features or chores. This is because we are delivering *Business Value*, not effort, and our budgets are in SPs, not effort. Effort is the variable here, since we are being agile.

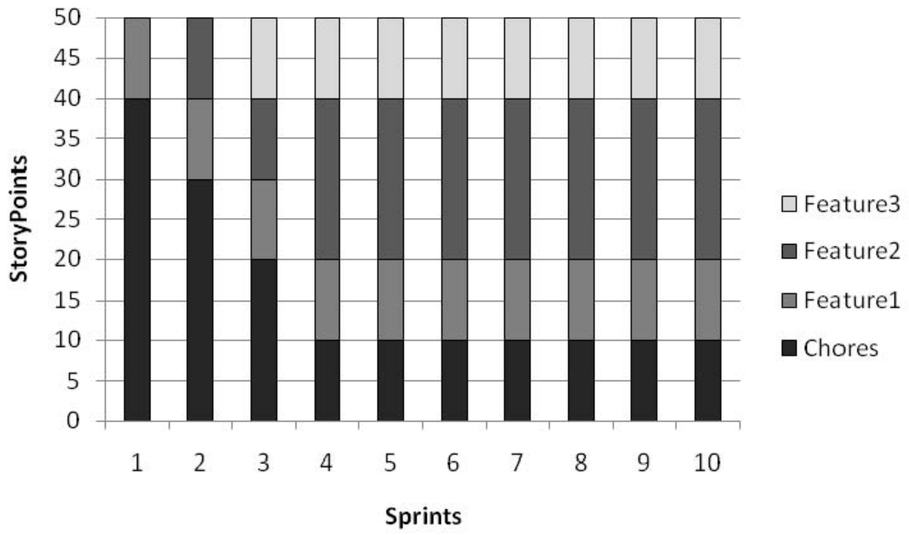
Let me note two things:

- Even though chores don't provide Business Value, their StoryPoints are included in the EVM metrics, as they represent value produced by the team.
- In my StoryPoint scale, a medium-sized story is worth 4 SPs, so these budgets range between 20 and 40 medium-sized stories.

  This is reasonable, since in my (use case-based) experience, it takes between 10 and 40 stories to produce a minimally releasable version of a (use case) feature.

Once we have the StoryPoint budgets, we make an "ideal" strategy for delivering them across the 10 sprints, as represented in the following graph. If the strategy worked perfectly, we would have the following delivery of SPs for the features, as we moved through the 10 sprints. Note that as we move through the stories for a given feature, we are assuming we're providing Business Value (for each feature) as if we were moving along the S-shaped graph given in Figure 1.

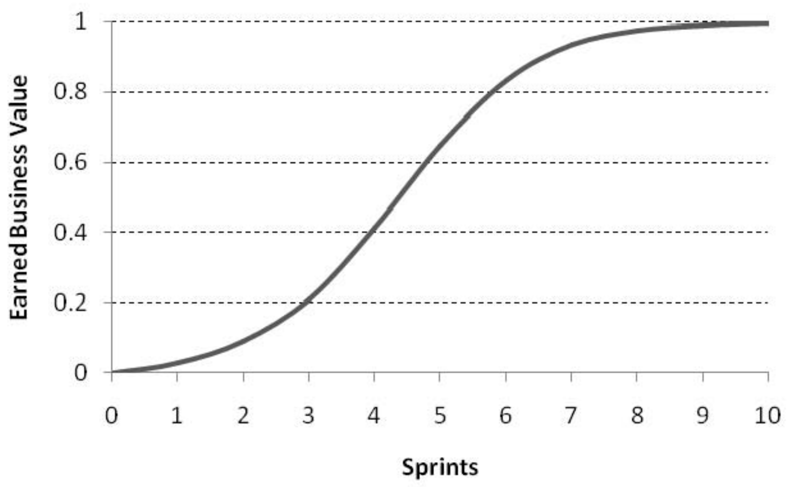
Figure 2: Ideal StoryPoint Delivery of the Release



It is very important to note that we *do not* know all the actual stories that will be delivered for the features at this point in planning. We may know some of them, but not all (and usually not even most). As part of the agile process, we are discovering them as we go through continuous discovery, analysis, and design. This continuous, reality-based, discovery process is inherent to agility, and results in us delivering Business Value in an optimized way, which we assume conforms to Figure 1 for each feature in this simulation. It is this process that allows the team to find a minimal releasable set of features as soon as it can, and then move beyond that point to provide additional Business Value. Typically, we think of this as getting the "meat and potatoes" out of the way first, and then adding in the "exciters."

The following figure gives the calculated EBV graph for this "Ideal Delivery" as we move through the sprints, based on earning BV as in Figure 1. Note that the presence of chores for the first few sprints flattens the curve compared to Figure 1, since the value chores produce is not BV.

Figure 3: The Project's EBV for the Ideal Delivery



Of course, there are also CPI and SPI graphs we could present, but in this ideal strategy they are both flat and = 1 all the time, as they are the ideal ones…

EXAMPLE OF METRICS IN NON-IDEAL SITUATION

Now, what happens when things don't go perfectly? What happens in real life? In particular, what happens if the CPI and SPI are both less than 1? That is, we are delivering fewer StoryPoints than we expected *and* they are more expensive than we expected.

We would think that we are in trouble, wouldn't we? Well, let's see what happens if we assume reasonable re-planning each sprint, as we should have in Scrum. In other words, our agile team maintained its poise and continued to look for and deliver the most important things it could find as it moved along.

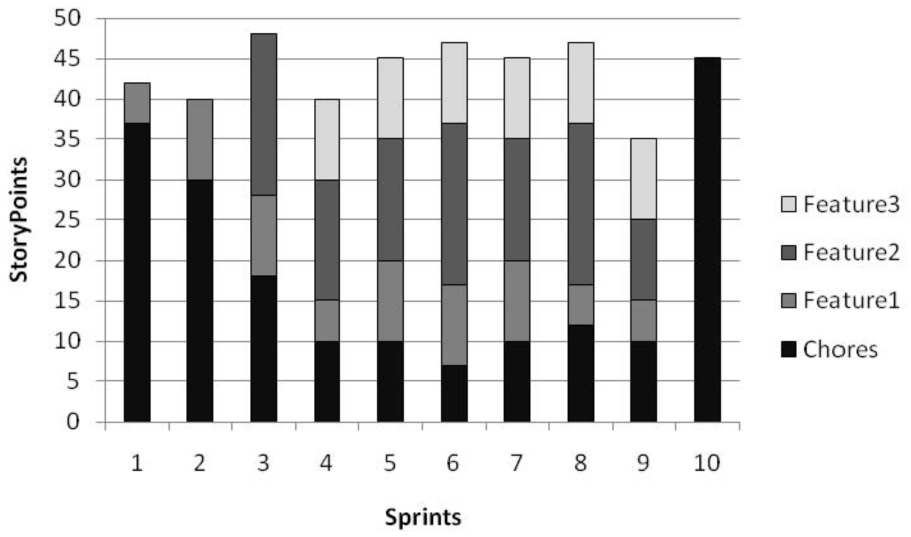
This is where the agile team either succeeds or fails. Does the pressure of being "behind" on CPI and SPI cause the team to lose its focus on what's important, and thus they don't even deliver good Business Value, or does the team maintain its focus and continue to do the best it can within its abilities? When the team is delivering less than it planned is when focus is most important, as they must find the most important stories, and be constantly balancing Business Value across features. Near the end of a release, it becomes crucial, as there is always some point when the team turns to the Product Owner and says something like, "You have about 20 stories left—make them good ones…"

First, let's look at what happens to our story delivery as things change and adaptation occurs. This adaptation results in a reordering (and possibly rethinking and/or repackaging) of the stories to be delivered in each sprint and is based on the realities that the team experiences and the search for Business Value and releaseability. The goal is to provide BV, and this is one of the primary realities that drive the re-planning.

In our simulated case, something else also happened. The team realized that integration was not as easy as they originally planned, so they set aside the *entire* last sprint for integration and hardening activities; that is, they decided the last sprint should be dedicated to "getting it out the door," and that it would, therefore, provide no additional Business Value. This, all by itself, decreased the expected BV-producing SPs by 40, which is a major hit.

The result was that the team did stories as in the following graph, where the last sprint has been completely turned over to chores (the 'getting it out the door' activities) and the other features were delivered as we see.

Figure 4: StoryPoint Delivery in a Typical (non-Ideal) Release



In order to complete the picture, we also need to know how much effort our team put in each sprint. Here is that data, given in person-days rather than currency.

Table 6: Person-Days Expended per Sprint

Sprint	1	2	3	4	5	6	7	8	9	10
Expected Days	50	50	50	50	50	50	50	50	50	50
Actual Days	50	52	65	50	55	60	50	55	50	55

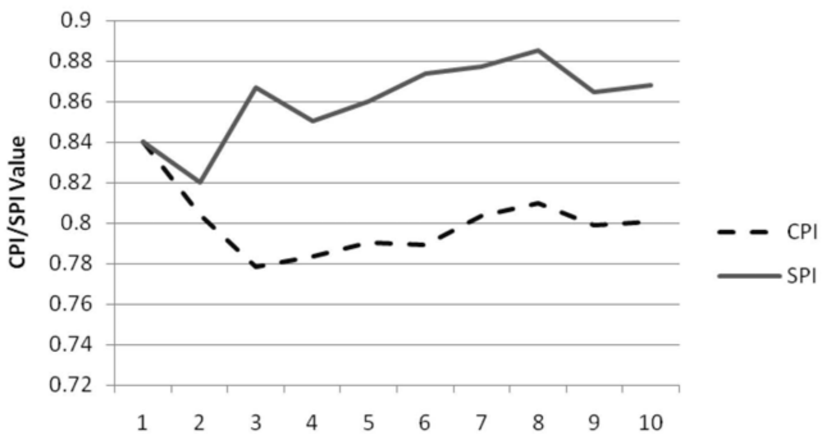
Now let's do some comparisons, and draw some graphs. First of all, we know we didn't deliver everything we wanted to, so let's look at what we did do on a feature-by-feature basis. Then we'll look at the graphics that showed the team's performance as it moved through the sprints.

Table 7: Comparison of Release Strategy to Actual Production

Feature	**Business Value (%)**		**Story Points**	
	Expected	*Actual*	*Expected*	*Actual*
Feature 1	0.5	.477	100	70
Feature 2	0.3	.288	160	115
Feature 3	0.2	.194	80	60
Chores	0.0	0.0	160	189
Totals	**1.0**	**.959**	**500**	**434**

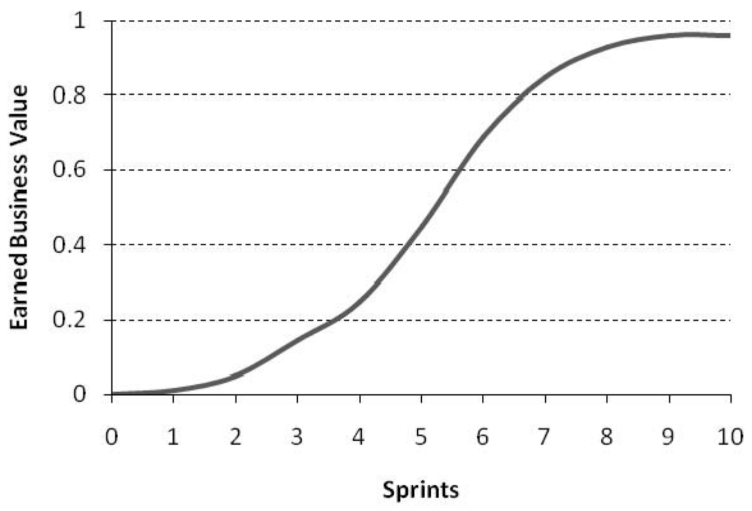
The total person-days used was 542 and the total SPs achieved is 434. Since we were planning for each to be 500, we're not surprised that we have the following CPI and SPI graphs.

Figure 5: CPI and SPI Graphs for this Non-Ideal Release



This looks bad, and it is—if what we're worried about is the delivery of SPs. Fortunately for the project, the re-planning at the sprint boundaries was done in a rational matter, with constant changes of the story delivery based on the story's BV. The result is that we wind up with the following Earned Business Value graph, which ends up with a maximum EBV of ~96% at the end of 10 sprints.

Figure 6: EBV Graph for this Non-Ideal Release



Note that even though we only delivered 87% of the SPs we wanted to (based on SPI), we got ~96% of the Business Value. Since we usually expect a minimal releasable product at about the 90% mark, we can call this project a success, as long as we're looking at Business Value. We didn't get everything we wanted, but we got what counted. This is actually not so surprising if we remember Pareto's 80/20 principle (or the 60/20 that we can actually achieve), but it is gratifying to see a simulation that demonstrates it so convincingly.

SUMMARY

Earned Value Management metrics are applicable to agile projects, but should be coupled with an Earned Business Value metric to give a complete picture.

Surprisingly (or perhaps not), EBV is not as tightly coupled to CPI and SPI as one might think.

Assuming a reasonable S-curve describing delivery of a feature's Business Value, we can still deliver a substantial overall Business Value even though we fall short on the "standard" AgileEVM metrics.

However, this only works if there is constant re-planning with Business Value as the main concern. The Product Owner must focus on delivery of Business Value and not the delivery of SPs. That is, the project must be agile…

ABOUT THE AUTHOR

Daniel Rawsthorne

Dan Rawsthorne, Ph.D. is a Certified Scrum Trainer and one of CollabNet's transformation coaches, helping organizations transition to Agile practices. Rawsthorne is a 27-year veteran of the software industry, having worked in a wide range of capacities, from coder to project manager, for companies large and small. In addition to possessing deep knowledge of many software processes, procedures, and techniques, he holds a Ph.D. in mathematics.

ABOUT COLLABNET

CollabNet is the leader in application lifecycle management (ALM) platforms for distributed software development teams. CollabNet TeamForge is the industry's most open ALM platform, supporting every environment, methodology, and technology. With an integrated suite of easy-to-use tools that share a centralized repository, it is the only ALM platform that enables a culture of collaboration, improving productivity 10-50% and reducing the cost of software development by up to 80%. As the founder of the open source Subversion project, the best version control and software configuration management (SCM) solution for distributed teams, collaborative development is in CollabNet's DNA Millions of users at more than 800 organizations, including Applied Biosystems, Capgemini, Deutsche Bank, Oracle, Reuters, and the U.S. Department of Defense, have transformed the way they develop software with CollabNet For more information, visit www.collab.net.

REFERENCES

1. Denis F. Cioffi, Ph.D., *New Tools for Project Managers: Evolution of S-Curve and Earned Value Formalism*, presented at Third Caribbean & Latin American Conference on Project Management, 21–23 May 2003
2. Sulaiman, Barton, Blackburn, *AgileEVM – earned Value Management in Scrum Projects,* presented at Agile2006, 23-28 July 2006
3. Rawsthorne, *Calculating Earned Business Value For An Agile Project*, AgileJournal.com, June 2006
4. Jeffries, Anderson, and Hendrickson: *Extreme Programming Installed*, Addison-Wesley, 2001.