[redacted] – Agile EV Chart

INTRODUCTION

EV, or earned value, has been a standard project metric put forth by the Project Management Institute, as a means of assessing project progress. This technique integrates scope, schedule, and resources, as a means of objectively measuring project performance/progress. Performance is measured by determining the budgeted (earned value) and comparing it to the actual work performed. Though this technique has traditionally been applied to projects using waterfall methodologies, Omnicell has endeavored to apply it to the agile methodology currently in use, with some modifications.

The EV chart in use today for the CSM project is constructed using a fixed number of story points – all stories deemed to be MVP for version 1 of the product – as well as a fixed end date – the last sprint beyond which there is a code freeze prior to GA – as a definition of project completion. Then, by dividing up the story points over the number of sprints until code freeze, a minimum velocity is obtained. Achieving (or averaging) this velocity over the life of the project would ensure completion of the assigned story points by this end date. However, because there are more than story points that go into product preparation, the current EV chart only shows part of the overall picture. Thus, a replacement or "improved" EV chart is being sought.

In researching a replacement for the traditional EV chart as a means of assessing project progress, I did the following:

- attended various webinars and online seminars on Agile Project Management,
- read various books and online publications (white papers, websites, etc.) dealing with Agile Project Management,
- researched various software PM tools and the associated reports and charts they produce, and
- met with Tommy Norman (who shares this MBO) to discuss options and alternatives.

Conclusions:

- The Agile methodology, by its nature, allows for constant change and re-assessment of the "end goal," so any chart that attempts to show progress with multiple fixed variables (a pre-determined delivery date, a pre-determined number of story points, a pre-defined fixed velocity, a fixed cost, etc.) is unrealistic and counter to agile tactics.
- Most "PM charts" in the agile world take the form of burn-down charts, iteration planning, or velocity trends, and thus do not accomplish Omnicell's goal for the EV chart – that being to provide management a sense of confidence in the project being on-track (and show when the project is veering off-track).
- We want to use actual data – currently housed in TFS – and be able to automate the production of the "new" EV chart as much as possible.

With this information and constraints, Tommy and I are proposing two charts:

- one which is an enhancement of the current EV chart, but shows bug points and code spike points as well as story points, and
- one which is a milestone forecasting chart, showing the desired goal and forecasted attainment of that goal, based on current trends.

Samples of these two charts are shown below [see S*ample Charts* section].

ANALYSIS

It is important to note the following aspects about the implementation of the new EV charts:

1. They will reflect only the software development activities present in the product backlog in TFS. Thus, other aspects of the project (hardware, manufacturing, marketing, beta activities, interdependencies with other projects, etc.) must be assessed and reported by other methods.
2. There is a burden on project participants to keep <u>all</u> data in TFS and to keep it accurate. This means the product backlog must be <u>prioritized</u> and <u>pointed</u> (assignment of points to stories, bugs, code spikes) in a <u>timely</u> manner.
3. Some processes need to be changed (discussed in the *Process* section below) to make the reports more reflective of reality and in so doing make the overall process more aligned with agile tactics.
4. The charts are merely a snapshot in time, and will vary from sprint to sprint. Because change is not constant – the product backlog will grow and shrink – no one chart can be viewed as providing definitive confidence in achieving project completion by a given end date; e.g., the charts are only an estimate based on current trends.

PROCESS

There is a concept of MVP (minimal viable product) which encompasses all the features desired for a particular version or release of an application. Currently, this has consisted of epics, themes, and stories in TFS with associated points and priorities. This has been considered a "fixed" variable. However, this has not taken into consideration some of the realities of software development. These are described below.

1. *<u>All</u> items in the product backlog must count,* including bugs and spikes. Bugs and spikes rely on the same personnel resources that are coding stories and so they impact story progress. Software is not "releasable" without bugs being addressed. Some stories are not "do-able" without some technical investigations (spikes) first. By counting only story points, we get a false sense of the amount of work there is to do. One can postulate that rushed or haphazard programming and testing (gather more story points and thus more perceived velocity) causes more bugs in the long-run, and slower more deliberate programming and testing might reduce bugs. When velocity takes into consideration both story points and bug points, one gets a more complete picture. By showing bug points on the new EV chart, we will show the portion of velocity spent on defects vs. stories.

2. *Points – whether for epics, themes, stories, bugs, or spikes – must be allowed to vary* each week to accurately reflect reality. <u>This does not mean that scope varies</u>! We still will interpret MVP to impose a fixed scope. Rather, the assessment of the effort required to do the work may change over time. For example, a story (epic or theme) may have originally been assigned X points, but as the story is attacked and elaborated upon in sprint planning, we find that the effort to accomplish the end result is greater or lesser than originally anticipated. Perhaps the story requires a spike first, or requires refactoring some existing code, or a change in architecture, or is more wide-reaching than originally thought. Perhaps portions of the story have already been accounted for in previously-played stories, or perhaps the function can be implemented in a more limited or conservative way and still satisfy the requirement.

3. *The entire backlog – stories, bugs, and spikes – must be prioritized on a similar scale,* at least up until a particular release or milestone (for example, GA for version 1.0). Furthermore, the backlog items must be prioritized similarly and timely, neither of which has historically been occurring. This is essential in order to produce intelligible EV charts and forecasts. If certain bugs are required to be fixed before beta, or certain stories are required to be played prior to beta2, these must be prioritized as such. For example, all items (stories, bugs and spikes) with priorities < 9000 are required for GA, all items required for beta have priorities < 4000, and so on. Having priorities that coincide with specific milestones or delivery dates allows for better assessment of whether the project performance is on-track, and assists the scrum masters in assigning backlog items in sprints in a timely fashion. Most importantly, it allows reports and the new EV charts to be produced "automatically" out of TFS.

IMPLEMENTATION

Steps to implement the improved EV chart and forecasting chart include:

1. Groom the product backlog.
   a. Categorize each item as story, spike or bug, so that it can be searched/reported on easily.
   b. Assign points to all items, knowing that the assigned points can change as the items get closer to being played.
   c. Prioritize all items *on a similar scale* and with milestones in mind. Currently, bugs have been prioritized with numbers of 1-9, while stories were prioritized with numbers of 1000-9000, which makes the sorting and assignment come up wrong.

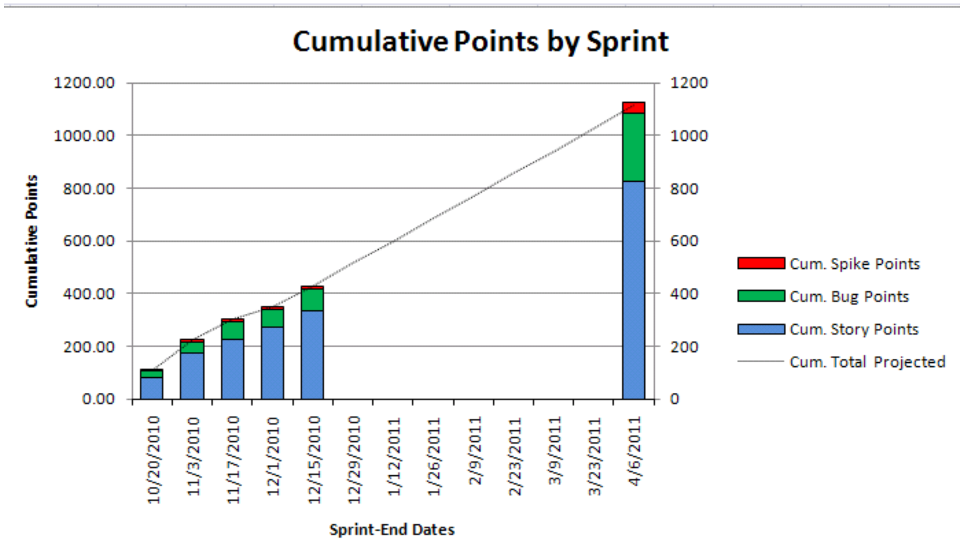
2. Work at least 1-2 sprints ahead, to assign the backlog items to sprints.
   a. Determine where the milestones coincide with certain sprints. You can hit a target with a higher degree of precision when it's closer than when it's farther away. Thus, the trend line for the intermediate milestones (events prior to GA) should be more accurate.

3. Create a report in TFS that can pull out the needed data easily. The charts should require little effort to update each week.
   a. Tommy has agreed to create the needed reports in TFS. For the time being, I will enter the reported data into an Excel spreadsheet that then draws the graph. This portion will not be automated until we move to TFS 2010.
   b. Realize that older historical data that was not categorized and/or pointed, will not be reported correctly. Thus, the new charts will only be accurate from about 10/20/10 forward.

SAMPLE CHARTS

Note: The sample charts shown below are mock-ups and do not represent actual data from the CSM project. They are for illustrative and discussion purposes only.

<u>EXAMPLE #1:</u>



The sample chart above is comprised of two parts: a stacked bar chart and a superimposed line graph. The stacked bar chart shows for each of 5 sprints (dates 10/20 thru 12/15) the cumulative points claimed on stories, bugs, and spikes. Thus, each sprint is a static snapshot in time, reporting what was actually accomplished. The very last sprint (date 4/6/2011) shows the total number of story points, bug points and spike points currently in TFS targeted to be completed by that milestone date. For GA, this includes all stories deemed MVP plus all bugs designated by the product owner to be fixed prior to GA. Thus, this last sprint is a "moving target" which will fluctuate as story points grow and contract, and as bugs are reported and spikes are needed.

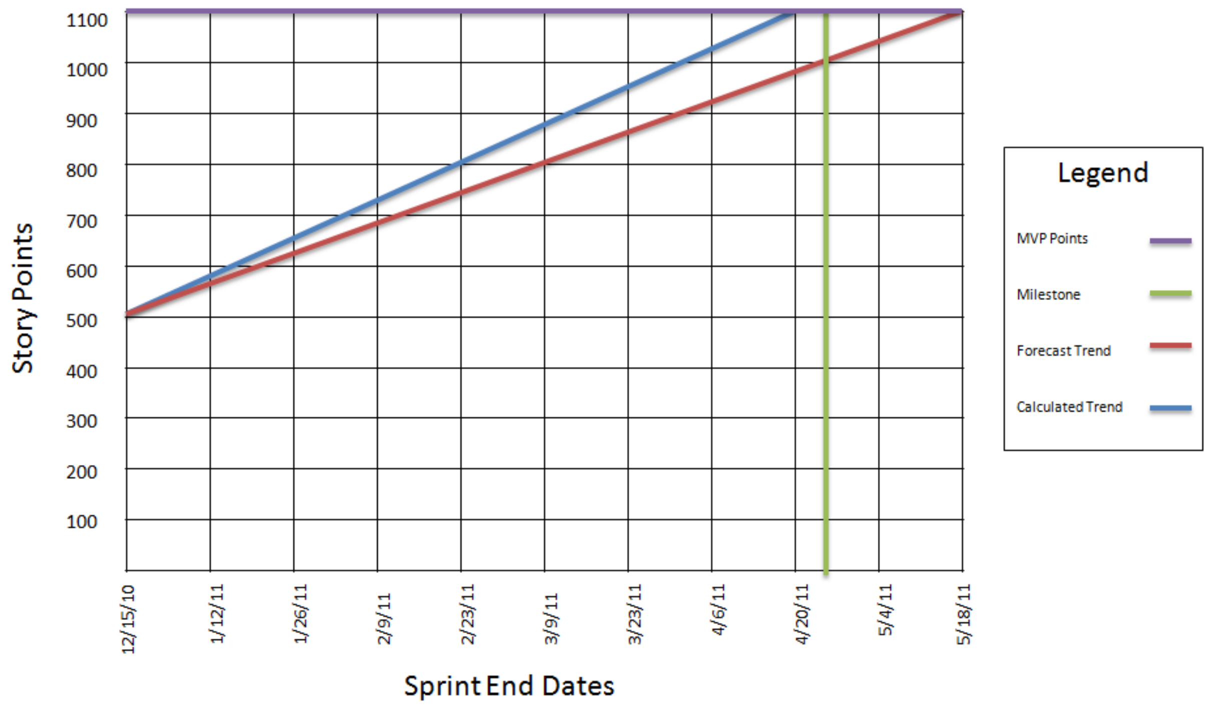
To assess whether the team can achieve the now changed target in the allotted time, we superimpose a line graph on the secondary axis. First, we calculated a "current average" velocity, based on the 6 most recent sprints, for total points (including bugs and spikes). Then, we project the line into the future and see where it crosses the milestone date. Thus the graphed line represents the cumulative number of total points (stories + bugs + spikes) that can be projected to be completed based on the current average velocity.

This chart is realistic and flexible. It takes into account that the backlog is constantly changing. It also takes into account that the team's velocity changes over time. It allows for any milestone date to be the "terminal sprint". In the above chart, we have selected the last sprint (code freeze) prior to GA. However, we just as easily could have selected the date of Beta1 or Beta2 as the milestone date, and then summed all the backlog items in TFS designated needed by that date (according to priority). So, for example, if GA includes all items with priorities < 9000, perhaps Beta2 includes all items having a priority of < 5000. The power inherent in this chart is only possible if ALL backlog items – stories, bugs and spikes – are prioritized and if they all use the same scale for prioritization.

<u>EXAMPLE #2:</u>

Milestone Forecast Report – Project Phoenix

There are **1100** total points (story, bug, and code spike) in the Product Backlog with a business priority of **6000** or less for the Minimal Viable Product (MVP). Based on the Scrum Team's forecasted average velocity of **70** points a sprint, they will complete 1100 points by the sprint ending on **5/18/11.** Based on the milestone date of **4/30/11**, the Scrum Team would need to reach the calculated velocity of **90** points per sprint to complete all the MVP points by the sprint ending closest to that date which is **4/20/11**.



The Milestone Forecasting Report is used to gauge if the total number of product backlog points (includes stories, bugs, and spikes) for MVP can be completed by a set milestone date. You enter the following parameters:

- Maximum Business Priority
  - This number is used to calculate the number of total points needed to reach MVP by summing the points for all product backlog items with a business priority less than or equal to the parameter value.
  - Using the same parameter the total number of completed points can also be calculated.
- End Date
  - This date is used as the starting point for the forecast.
  - This will default to the end date of the current sprint in TFS.
- Sprint Length
  - The number of days in each sprint is used to calculate the report axis containing the sprint end dates.
  - This will default to the length of the last sprint in TFS.
- Forecasted Average Velocity
  - This metric is used to calculate how many sprints it will take to complete the remainder of the MVP product backlog points.
  - This will default to the average velocity for the last 6 sprints.
- Milestone Date
  - This date is used for calculating the velocity required by the team to finish all the MVP by the entered date.

From these parameters this report will pull live data from TFS to produce the above report. The purple line at the top of the graph is the total MVP points (all product backlog items) below a specified business priority level. The green line is the milestone date. The blue line is the calculated trend for finishing all MVP points by the milestone date. The red line is the trend showing the actual date (based from the start date parameter) the team would complete the MVP points based on their forecasted velocity.
When the red and blue trend lines are identical then the project is on track to finish on the milestone date. If the calculated trend is different, you can play “what if” scenarios by changing the parameters and seeing what can be modified to bring them together again:

- Scope – Changing the maximum business priority will change the MVP points.
- Time – Changing the milestone date will allow you to see how much can be completed by a certain point in time.
- Velocity – Changing the forecasted velocity allows you to see how much an increase (or decrease) in the teams average velocity would affect your actual completion date.