Technical/Control Account Manager Guide

Obsolete

This material is obsolete. Please refer to DMTN-020 for up-to-date information.

Unknown macro: 'redirect'

Introduction

Each group participating in DM is coordinated by a both a scientific lead and a technical manager. The technical managers are expected to play the role of Technical/Control Account Manager (T/CAM), ensuring that activities in their group are properly accounted for according to the the Earned Value Management System (EVMS) used by LSST. Mapping the agile software development practices used within DM to the EVMS structure can be complex, and the procedure will likely evolve with time. This page attempts to summarize accumulated wisdom & best practice. It represents the understanding of the author(s), which is not necessarily correct and cannot be comprehensive; in case of questions, please defer to the word of the DM Project Manager, DM Project Scientist or LSST Project Controls Specialist – and then update this page to reflect their advice.

Jargon

- **CAM**: Control Account Manager.
- **EVMS**: Earned Value Management System.
- **MREFC**: Major Research Equipment and Facilities Construction.
- **OBS**: Organizational Breakdown Structure.
- **PMCS**: Project Management Control System. The suite of tools ultimately used for reporting progress to the outside world.
- **Primavera**: An important component of the PMCS suite.
- **WBS**: Work Breakdown Structure. All work items are hierarchically organized into the WBS tree; CAMs are responsible for one or more WBS elements.

Planning process & interaction with JIRA

The high-level DM development roadmap is defined at a yearly resolution in document LDM-240. This sets the broad goals for each WBS for each year.

In practice, development is scheduled in 6 month long cycles, labelled "Winter xx" and "Summer xx", for year xx. At the start of each cycle, T/CAMs must agree with the DM Project Manager and Project Scientist the work which their group will undertake over the following 6 months (but NB that setting the priorities which define the order in which work should be addressed is ultimately the responsibility of the DM Project Scientist and per-institution science leads). This information is captured in the form of "epics". An epic defines a large but self-contained piece of functionality, which has a start date, a well defined end goal and set of acceptance criteria. Each epic is identified with a particular WBS element. Ideally, epics encompass about one month of work for a single developer; longer epics may be appropriate in some cases, but epics longer than 3 months should be split into multiple epics.

When defining epics, we estimate the amount of work that is required to complete them. This estimate is made in terms of "story points". A story point corresponds to one half day of uninterrupted work by a competent developer. It is important to remember that the estimated story points of an individual epic or story are not velocity-adjusted; they are the "ideal" amount of work that needs to be done by a competent developer. This is the number to put into JIRA, the velocity adjustment occurs when the epics are imported to PMCS, and in sprint planning only.

In practice, of course, uninterrupted work is not the normal state, nor is the set of epics and stories in JIRA the only activities a person may be engaged in o n the project. For example, certain people are in leadership positions, and must attend non-DM project meetings and reviews. These other activities, while not in JIRA, are accounted for in PMCS. Without those elements taken into account, a developer would have roughly 5 days/week * 26 weeks * 2 SP/day = 260 SP available in a 6 month cycle.

Taking the other activities and interruptions into account, the total number of story points that a developer can actually complete in all the sprints in a 6 month cycle is estimated as:

Developer, no science time, no leadership resonsibility	157.5
Developer, no science time, leadership responsibility	67.5
Scientist, 20% science time, no leadership responsibility	112.5
Scientist, 20% science time, leadership responsibility	22.5

(In fact, these estimates are calculated by assuming an efficiency factor of ~0.7, but *you do not need to know that* and that factor could be revised, or even changed for particular groups, in future. The spreadsheet which these calculations are extracted from is attached to this page). Obvious corrections can be made for shorter cycles (e.g. Winter 2015, which is a 5 month cycle) or to calculate the number of story points which can be addressed per sprint.

Epics must be resource loaded (i.e. people assigned to work on them at a certain level) and scheduled in PMCS (i.e. the system must know when each epic is due to start and due to end). To make this possible, when submitting epics to the DM Project manager, it is necessary to specify which developer(s) will be undertaking the work (fractional allocations along the lines of "this epic will be 40% handled by X and 60% by Y" are fine) and the order in which epics are to be completed. The DM Project Manager uses this information to calculate a complete schedule for the cycle.

Epics as entered in PMCS correspond directly to epics defined in the JIRA issue tracking system. In fact, the epics in JIRA are imported into PMCS as "activities". Within JIRA, epics may be created at any time; groups may wish to define a large number of epics covering their work into the future. However, at the start of a cycle, all epics to be handled in that cycle must be appropriately tagged with the cycle name and their WBS, as well as having story points estimated and developers assigned. The story points estimate does not have to be derived from a complete list of stories at this point, and in most cases that list will not yet exist since stories emerge over time in the Agile process. Whether an explicit list of stories is created at this time or not, analysis and experience should guide the epic story point estimate. Ultimately, the actual stories will determine the work performed in the epic, but the PMCS and EVMS require an initial estimate at this point for scheduling and resource loading.

Start and end dates for epics are not currently specified within JIRA but this is under consideration. (JK - Why not use Planned Start and End for this?)

Within JIRA, epics are further broken down into "stories". A story is a small but self-contained piece of work, ideally corresponding to no more than a few days of developer effort. Our rule of thumb is that stories should be at least 2 SP and absolutely no more than 20 SP. If a Story is more than 20 SP, it should be divided into multiple stories. Typically, a series of stories are defined for an epic before work on that epic begins, however this need not be communicated to the DM Project Manager. Stories have their own story point allocation and assigned developer. Note that it is not necessary that the sum of points allocated to all stories in an epic is equal to the number of points estimated for that epic (but see later); similarly, it is not required that the assignees for the stories are the same as the original developers communicated to the Project Manager.

As development progresses, stories are assigned to epics and the progress through the stories associated with the epics defined in PMCS is estimated by recording which of the constituent stories of those epics have been marked as "done" using JIRA. Note that this is a binary state: stories which are partially complete, in review, reviewed, etc count for no value earned; stories which are complete count for 100% of their story point value. For this reason, it is essential that developers mark stories as done on JIRA when they are complete. The LSST Project Controls Specialist imports the updated story status on a weekly basis to keep the PMCS status updated.

It is acknowledged that drawing up a comprehensive development plan for the next 6 months at the start of a cycle is challenging. However, after the epics for a cycle have been entered in PMCS, it is not possible change them without going through a formal process. After 3 months – ie, at the halfway point of a development cycle – there will be an opportunity to "reset" and change the epics planned for the rest of the cycle.

There is significant flexibility in the system in spite of this. This is because while epics are change controlled, the stories associated with them are not. Developers can perform the stories in and epic any order, and can add, delete, or move stories between epics, and the changes will be detected on importation to PMCS. There are a number of techniques used by the LSST Project Controls Specialist to keep the % complete in PMCS from moving backward and to keep the epic estimate aligned, as the stories are changed.

Handling "emergent" work

The planning process described above assumes that work can be schedule 6 months in advance. In practice, of course, this is not always the case: bugs will occur, and feature requirements that were not anticipated at the original design stage will emerge. Through careful planning, it should be possible to minmize this unexpected work, but it is not plausible to suggest that it will be completely eliminated. We therefore adopt the following procedure.

At the start of a cycle, it is appropriate to allocate some fraction of the total story point budget to unfilled "bucket" epics. Since it is important to minimize unplanned work, the fraction should be small: 10% is a useful guideline. As bugs and new features are discovered, they can be added to JIRA using the issue types "bug" and "improvement" respectively. These can then be added to the bucket epics defined and worked on during the cycle.

Of course, not all bugs and features require an urgent solution, and some may involve substantial amounts of work. In this case, it is better to delay the work until the next cycle, and, if it is substantial enough, create a new epic to describe the requirements. There is no hard boundary between emergent work which can be added to a bucket in the current cycle and that which should be held over until the next cycle and planned in from the start: this is a management decision which will depend on the case in question. However, as above, bear in mind that the amount of emergent work tackled in a cycle should be minimized where possible.

(Note that there has been some historical debate about the semantics of the "bug" and "improvement" issue types in JIRA and questions over whether they "count" in an earned value sense. As of April 2015, this has been resolved: see RFC-43 for details.)

Claiming earned value

It is worth summarizing the requirements for work to be regarded as "value earned" in PMCS. These are:

- The work must be described by a JIRA issue in the DM project with the type set to "story", "bug" or "improvement";
- The issue should be labelled with a story point value appropriate to the amount of time the work is expected to take;
- The issue should be part of a JIRA epic;
- The epic should be tagged for work during the current cycle (using the "Cycle" field) under the appropriate WBS ("WBS" field).

Tools

JIRA interaction

The lsst-sqre/jira-tools repository on GitHub contains some Python-based tools for interacting with JIRA and calculating quantities (such as the total number of points assigned to all the constituent stories of an epic) which are not straightforward to determine from the web interface. Improvements & contributions to this code by means of GitHub pull request are welcome.

Submitting plans to the Project Controls Specialist

When submitting your plan to the Project Controls Specialist before the start of a new cycle, the important information to capture includes:

1. The fraction of their total time that each member of your staff will spend working on LSST over the cycle;
2. The amount of time each member of staff in a leadership role will spend on level-of-effort management activities;
3. The relative allocation of their remaining time between epics.

Point 3 above can (& will) be derived from the relative allocation of planned story points to epics on JIRA.

This Excel spreadsheet captures the information required; please use it to submit your plans.

Long Range Planning (LDM-240)

Data Management maintains a roadmap for construction in document LDM-240 (see Project Planning for Software Development). This roadmap was originally created as a spreadsheet in the Design Phase, and later migrated to JIRA Agile during construction. This document is now generated by adding Milestones, Meta-Epics, and Key Performance Metrics in JIRA Agile in the Data Management Planning Project (DLP), and linking Epics from the Data Management Project (DM) to the Meta-Epics and Key Performance Metrics. The schema for both projects with the associate linkages is contained in dm planning diagrams.pdf.

FAQ

What is "level of effort"?

"Level of effort" work is scheduled activity which is not tracked through JIRA. Examples include planning work undertaken as a T/CAM. It is discounted against the number of hours available for working on story points. For example, see the calculation on page 3 of the DMLT November 2014 W15 status report.

What is a "monthly narrative"?

For every (calendar) month that passes, the T/CAM must supply the Project Manager with a description of the work carried out by their group during that month. Since the list of epics & stories which were completed during this month can be automatically extracted from JIRA, this report should go beyond simply that list. Rather, it should describe in human-readable terms the work undertaken and how it relates to the overall plan.

The report should be broken down by WBS. List planning/hiring/LOE activities under:

> 02C.[your overall WBS].00 [Your area] Management Engineering and Integration

For example:

> 02C.04.00 Data Release Production Management Engineering and Integration

Please provide the following information on recruiting and hiring (text, not just numbers):

- Positions opened
- Interviews conducted
- Offers made
- Positions filled
- Remaining positions unfilled

Note that the monthly narrative is sometimes also referred to as the "technical progress report" or TPR. Refer to the Project Planning for Software Development page for more information about reporting.

What if the number of points estimated for an epic doesn't correspond to the total of the points in its constituent stories?

It is possible that when an epic is fleshed out with stories their total number of points will not be equivalent to the number of originally estimated for the epic. That's fine: the original number of points was used to defined the duration of the epic in PMCS, but the number of points per story will be used in defining its percentage completion. This may lead to an over- or under-run on the completion date or resource allocation, and that may ultimately need to be officially explained in addition to the normal monthly narrative. The Project Controls Specialist produces are EV variance report each month, and indicates any variances that require such explanation.

More complex is when an epic has been partially completed and then it becomes clear that unanticipated complexities mean more work will need to be done to complete it properly. More stories are added to the epic, and a naive story point based calculation would show its percentage completion to *decrea se*. Going backwards is not encouraged, so this would not be a good thing. Instead, the percentage completion will be held constant, by effectively down-weighting the new stories being added in comparison to the work already completed. This is handled in conjunction with the Project Manager and/or Project Controls Specialist.

How do we account for some developers being more efficient than others?

For example, if a particular story is assigned to a developer known to work quickly, should its number of story points be reduced to reflect the rate at which they might complete it?

The answer, in brlef, is: no. The story points should be an estimated based on the performance of an average developer. If changing developers caused the number of story points to change, it would imply that slow developers earn more value for the project, which would be (at best) misleading.

We have found this approach to be the most feasible over the entire DM effort. Tracking the individual productivity of each of 45 DM staff would be difficult and error-prone. The net performance of teams within DM is more feasible and yields more stable results.

What happens to stories marked "won't fix"?

We leave them as story points in Primavera, but we zero out the EV weight in PMCS so they don't count towards progress measurements.

Scope for overheads

At the start of a cycle, it is likely impossible to accurately predict all the steps that are necessary to complete a given epic – even if considerable planning effort is devoted to it and a set of stories are fleshed out, small adjustments to the plan will likely be necessary in the light of real world experience. In addition, bugs may arise which block the completion of a particular epic and/or have major impacts on the project as a whole, and so must be addressed. It is therefore appropriate to build some overhead into the plan to account for these. A **small** percentage overhead may be included in addition to the estimated story points for an epic to account for unanticipated issues which must be addressed to complete the work. For example, if the total points planned for stories in an epic is 50, it may be appropriate to allocate 55 story points to complete the epic.

Resource Loading

At the start of a cycle, after the required planning information has been supplied to the Project Controls Specialist, they will produce a summary of the resource loading for this cycle. This shows which of your staff are working on what when. A few words of explanation may be helpful:

"Budgeted units" refers to "working hours". It is assumed that a full-time software developer on the project has 1800 working hours per year, while a full-time scientist has 1440 (the remainder being their personal science time). Staff working on the project part time have their working hours reduced in proportion. When considering the "budgeted units" for each staff member in the resource loading spreadsheet, you should expect to see half this number (ie, a developer should be budgeted for 900 units in a cycle, a scientist for 720). *No other overheads are taken into account in this resource loading.* For example, in the discussion above, we assumed a 70% velocity when considering story points: this is *not* relevant when considering budgeted units.