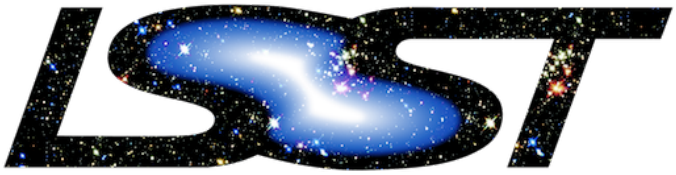

LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)

Data Management Project Management Guide

Jacek Becla, Frossie Economou, Margaret Gelman, Jeff Kantor, Simon Krughoff, Kevin Long, Fritz Mueller, William O'Mullane, John D. Swinbank, Xiuqin Wu

DMTN-020

Latest Revision: 2018-02-20

Abstract

This is the DM guide for T/CAMs implementing the earned value system.

Change Record

Version	Date	Description	Owner name
1	2016-07-01	First release	JB et al.
2	2016-11-26	Add section on software releases	JDS
3	2016-11-28	Add material on relationships between milestones and the appropriate WBS for bucket epics	JDS
4	2018-02-09	Reflect new practices. Convert to LaTeX.	WOM, JDS
4.1	2018-02-20	Policy on Jira/Confluence for non DM people in Section 13.2 (DM-7675).	WOM

Contents

1 Introduction **1**

2 Important Documents **1**

3 Useful Contacts **2**

4 Technical Managers **2**

5 Formal Organizational Structure **2**

5.1 Work Breakdown Structure 2

5.2 Organization Breakdown Structure 3

5.3 The Control Account Manager 4

6 Earned Value Principles **4**

6.1 Labor Costs 5

6.2 Variance Narratives 5

6.3 Level of Effort Work 6

7 Estimating Effort **6**

7.1 Basic Assumptions 6

7.2 Special Cases 8

7.2.1 Newcomers 8

7.2.2 Technical Managers and other Leadership Roles 8

7.2.3 Institutional Science/Engineering Leads 8

8 Long Term Planning **9**

8.1 Planning Research Work 10

8.2 Earned Value and Planning Packages 10

8.3 Epic-Based Long Term Plans 11

8.4 Software Releases 12

9 Short Term Planning **12**
9.1 Cycle Cadence & Release Planning . . . 12
9.2 Defining The Plan . . . 13
9.2.1 Scoping Work . . . 13
9.2.2 Defining Epics . . . 14
9.2.3 Scheduling Research Work . . . 16
9.2.4 Bucket Epics . . . 16
9.2.5 Mapping SPs to BCWS . . . 17
9.2.6 Cross Team Work . . . 17
9.3 Revising the Plan . . . 18
9.4 Closing the Cycle . . . 19

10 Execution **19**
10.1 Defining Stories . . . 19
10.2 Sprinting . . . 20
10.3 Completing Epics . . . 21
10.4 Handling Bugs & Emergent Work . . . 22
10.4.1 Receiving Bug Reports . . . 22
10.4.2 Issue Types . . . 23
10.4.3 Scheduling . . . 23
10.4.4 Relationship to Earned Value . . . 24
10.5 Earning Value . . . 24
10.6 Jira Maintenance . . . 25
10.7 Coordination Standup . . . 25
10.8 Monthly Progress Narratives . . . 26

11 Reporting Actuals **26**

12 Standard Reporting Cycle **27**

13 Personnel **28**
13.1 Staffing Changes . . . 28

13.2 Collaborators 28

14 Glossary **28**

15 References **30**

Data Management Project Management Guide

1 Introduction

This document provides a guide to the mechanisms underpinning LSST Data Management's approach to project management. It is intended to be read in conjunction with LDM-294, which describes the organization and management of the DM subsystem, and with reference to LPM-98, which describes the LSST Project Management Control System (PMCS).

2 Important Documents

Wherever a conflict arises, baselined project documentation takes precedence over this note. You are encouraged to submit bug reports so that this document can be made compliant.

Be aware of prefixes: "LDM-" documents refer specifically to the Data Management subsystem, "LSE-" to Systems Engineering, "LPM-" to Project Management.

LDM-294 *Data Management Organization and Management* Describes the overall structure of LSST Data Management, the goals of the project, the approach being taken to reach them, and the roles of the various individuals and groups involved.

LPM-43, LPM-44 *Work Breakdown Structure (WBS)* and *WBS Dictionary*, respectively. The former shows the overall work breakdown structure for the whole project. Note that these documents are periodically extracted from the master PMCS, and therefore occasionally do not reflect the most recent changes; LDM-294 may be a more useful everyday reference.

LPM-98 *LSST Project Controls System Description*. Describes and defines the components of the PMCS used to manage and report on the overall LSST Project.

In addition, you should be familiar with the EVMS Training – CAM 101 slides presented by the Project Controls Specialist (§3) at the LSST 2014 Meeting.

3 Useful Contacts

The LSST DM Project Manager is William O'Mullane and the Deputy PM is John Swinbank. They are the first point of contact for all issues regarding project management within DM. The Subsystem Scientist is Mario Juric who is the first point of contact for science or scientist related questions.

The LSST Project Controls Specialist is Kevin Long. He is responsible for the PMCS and, in particular, for ensuring that DM properly complies with our earned value management requirements. He is the first point of contact for all questions regarding PMCS.

4 Technical Managers

This guide is primarily aimed at the LSST DM technical managers. Technical managers report directly to the DM Project Manager. Technical managers are, in general, expected to act as Control Account Manager (CAM) and technical lead for their WBS elements; as such, they are sometimes referred to as "T/CAMs".

The T/CAM role is described in LDM-294, while the specific duties of a CAM are discussed in detail in §5.

5 Formal Organizational Structure

5.1 Work Breakdown Structure

The LSST WBS is defined in LPM-43 (see also LPM-44 for an extended—but not universally illuminating—definition of what each level of the breakdown consists of).

The WBS provides a hierarchical index of all hardware, software, services, and other deliverables which are required to complete the LSST Project. It consists of alphanumeric strings separated by periods. The first component is always "1", referring the LSST Construction Project. "02C" in the second component corresponds to Data Management Construction. Subdivisions thereof are indicated by further digits. Subdivisions at this level correspond to teams within the DM project. Thus:

WBS	Description	Lead Institution

WBS	Description	Lead Institution
1.02C.01	System Management	LSST
1.02C.02	Systems Engineering	LSST
1.02C.03	Alert Production	University of Washington
1.02C.04	Data Release Production	Princeton University
1.02C.05	Science User Interface	Caltech IPAC
1.02C.06	Science Data Archive	SLAC
1.02C.07	Processing Control & Site Infrastructure	NCSA
1.02C.08	International Communications. & Base Site	NCSA & LSST
1.02C.09	Systems Integration & Test	LSST
1.02C.10	Science Quality & Reliability Engineering	LSST

These subdivisions are referred to as the *third level WBS*. Often, they are quoted without the leading "1" (e.g. "02C.01"), but, even in this form, they are referred to as "third level".

All of these third level WBS elements are subdivided, forming a fourth level. The fourth level always contains a "00" element, which is used to capture management and Level of Effort (LOE) work, and may contain other fourth level, or even deeper, structure. Nodes in the WBS tree are referred to as elements.

5.2 Organization Breakdown Structure

In parallel with the WBS, we have an Organizational Breakdown Structure (OBS), which assigns each institution involved in the project a unique numeric identifier. The OBS is defined in LPM-98. Those institutions directly relevant to DM include:

OBS	Institution
1.01	LSST
1.02	SLAC
1.03	Caltech IPAC
1.04	NCSA
1.05	University of Washington
1.06	Princeton University

5.3 The Control Account Manager

A control account is the intersection between the WBS and the OBS. Each control account falls under the purview of a CAM. Typically within DM, a single CAM is responsible for the whole of a third level WBS. That is, the manager at the lead institution for a particular component is responsible for all work performed on that WBS element, even if some of that work is performed at another institution.

6 Earned Value Principles

LSST DM is funded by as an National Science Foundation (NSF) Major Research Equipment and Facilities Construction (MREFC) project. Under the terms of the MREFC award, we are required to follow an *earned value* approach to project management. A full description of the earned value approach is outside the scope of this document: the project will provide formal training. We provide a brief aide-mémoire for convenience only.

The earned value technique assigns each component of the system with a dollar value corresponding to its expected cost of production. In a (largely) software based project like LSST DM, it is often convenient to equate the cost of production with the cost of the labor required to write the code: in the more general case, however, it also includes cost of hardware procurements, etc. This provides a convenient heuristic for estimating cost: given some nominal labor costs, the cost of a component is a proxy for the amount of labor required to produce it.

As well as a cost, the plan includes a start date and a completion date for each component.

The total value of work which *should* have been completed by a particular date is the Budgeted Cost of Work Scheduled (BCWS). The total value of work which has *actually* been completed by the date is the Budgeted Cost of Work Performed (BCWP). The total sum expended on the work is the Actual Cost of Work Performed (ACWP). Theoretically, if estimates of both cost and time for every component of the system are accurate, at the end of construction, all of these three quantities will be equal.

In practice, estimation is rarely perfect. Imperfect estimates are exposed as variances. Specifically, we can show either Schedule Variance (SV)—a negative value means that less of the system has been delivered to date than planned—or Cost Variance (CV)—a negative value means that the work delivered to date has been more expensive than predicted. Related

quantities, Schedule Performance Index (SPI) and Cost Performance Index (CPI), express the same information as ratios rather than sums.

In the limit of perfect planning, variances will be zero. While this is our aspiration, we recognize that is a practical impossibility: variances are perfectly normal in a project, and we should not be afraid to have them as long as we can clearly explain their cause and the steps we are taking to mitigate them (see §6.2).

Variance is perfectly normal in a project and we should not be afraid to have them and provide narratives about them. We should take care the narrative is not always the same i.e. if we have a negative variance every month and it is because we did not plan for something we should do better planning.

All of these indices can be applied to any WBS element within the project. Thus, we can talk about value earned across the whole of DM (1.02C) or on a specific component (say, the User Workspace Toolkit, 1.02C.05.05).

6.1 Labor Costs

Our methodology is designed to avoid exposing individual salaries to the wider project. Therefore, when calculating labor costs for earned value purposes, we do not rely on a known cost per individual. Instead, all staff are assigned to one of a number of types (typically within DM we use scientist, senior scientist, developer or senior developer, but there are a several alternatives available at the project level: see LPM-81 table 5-2 for the full list), each of which is assigned a nominal cost level according to institution: it does not vary between individuals of the same type within the same institution. This nominal cost does not, therefore, correspond to a particular individual, but is a broadly defined expectation. Full details are available in LPM-81.

6.2 Variance Narratives

Every month, the eCAM tool is updated from the PMCS to reflect the latest earned value status. If either cost or schedule is behind schedule by more than either $100,000 or 10% you are required to provide a “narrative”. This is divided into two parts: you must explain why the variance arose, and what action will be taken to correct it (e.g. slipping work into the future, or diverting resources from elsewhere to make up the shortfall). The narrative is entered directly into eCAM.

In future, narratives may also be required for positive variances (i.e. running ahead of schedule).

Variance is calculated on a monthly basis; variance narratives are due in the second week of the calendar month following that to which they apply (refer to §12 for details). Note that the variance narrative should not regularly be the same from month to month: if the same circumstances keep recurring, it is indicative of a fault in our planning which we must address.

6.3 Level of Effort Work

The implicit assumption in the earned value technique outlined above is that all work corresponds to a specific deliverable. However, parts of our work do not: every member of the team will find it necessary to attend meetings or take part in other activities which do not directly map to deployed code. This may be particularly the case for technical managers or others in leadership roles within the project. This work is referred to as LOE: it is assumed to earn value simply through the passage of time.

In general, we strive to minimize the fraction of our effort which is devoted to LOE activities and favor those which are more directly accountable. However, in certain cases such as operations of pipelines or other systems, LOE is perfectly acceptable.

Our first-order estimate is that developers will spent 30% of their time on LOE type activities, and the remaining 70% of their effort is tracked against concrete deliverables. However, as above, we generally aspire to minimize the fraction of LOE: T/CAMs are therefore encouraged to explicitly schedule as much developer time as is possible.

Note that all LOE work should be invoiced to the “00” fourth-level WBS element (1.02C.03.00, 1.02C.04.00, etc)

7 Estimating Effort

7.1 Basic Assumptions

The Project assumes that a full-time individual works for a total of 1,800 hours per year: this figure is *after* all vacations, sick leave, etc are taken into account. Staff appointed to “developer” positions are expected to devote this effort directly to LSST.

Appointment as a "scientist" includes a 20% personal research time allowance. That is, scientists are expected to devote 1,440 hours per year to LSST, and the remainder of their time to personal research.

Personal research time is *not* chargeable to LSST under any WBS or account, including level of effort. The Project expects to pay the full rate for an individual with research time who contributes 1,440 hours to the project, and does not require any accounting of the remaining 360 hours.

When reporting actual costs (§11), it may be helpful to consider the following examples:

A developer for which the total annual cost (salary, overheads, etc) is A charges an hourly rate of $A/1800$.

A scientist with total annual cost B charges an hourly rate of $B/1440$.

No further corrections are necessary. In particular, there is no difference in the way working hours are measured, or the conversion of Story Points(SPs) to hours.

Our base assumption is that 30% of an individual's LSST time (i.e. 540 hours/year for a developer, 432 hours/year for a scientist) are devoted to overhead for regular meetings[1], ad-hoc discussions and other interruptions. This work is counted as LOE (and, as such, is charged to the relevant "00" fourth level WBS element, as described in §6.3). It is actively encouraged to allocate less than 30% of an individuals time to LOE where that is possible.

Assuming no variation throughout the year, we therefore expect 105 hours of productive work from a developer, or 84 hours from a scientist, per month. Note that this is averaged across the year: some months, such as those containing major holidays, will naturally involve less working time than others: the remainder will necessarily include more working time to compensate.

Rather than working in hours, our Jira based system uses SPs, with one SP being defined as equivalent to four hours of effort by a competent developer. Thus, we expect developers and scientists to produce 26.25 and 21 SPs per *average* month respectively. This is summarized in Table 4.

[1] "Meetings" include, for example, scheduled weekly team meetings, stand-ups, etc; major conferences or project meetings involving preparation, travel time, etc should be scheduled in advance and allocated SPs.

	Hours		SPs
	Per year	Per month	Per month
Developer	1800	105	26.25
Scientist	1440	84	21.00

TABLE 4: Expected working rates for developers and scientists.

7.2 Special Cases

7.2.1 Newcomers

New or inexperienced developers, even when devoting their full attention to story-pointed work, will likely be less productive than their more experienced peers. In this case, the ratio of hours to SPs increases, but the number of hours remains constant.

Note that specific activities related on "onboarding" and getting up to speed with the project can be ticketed as regular work. For example, working through tutorials, reading documentation, and so on are all activities which can earn SPs. Typically, this will assigned to the "00" (management) element of the WBS (§5.1).

7.2.2 Technical Managers and other Leadership Roles

Individuals in leadership roles may find it necessary to assign a larger fraction of their time to LOE type work, and therefore spend fewer hours generating SPs. The ratio of hours to SPs remains constant, but the number of hours decreases.

7.2.3 Institutional Science/Engineering Leads

As described in LDM-294, as well as a T/CAM each institution with DM has a Science/Engineering Lead. This role is not equivalent to being granted personal research time, but reflect a level of scientific oversight within the project: they act as the overall Product Owner for the parts of the system that their institution has been tasked to deliver. While the Science Lead reports to the Subsystem Scientist, they primarily provide a service to the local T/CAM, and, as such, are charged to a budget corresponding to the institutional WBS (and *not* to e.g. DM Project Science). Time spent performing this role must be accounted for in the usual way (either as LOE or as providing deliverables, preferring the latter whenever possible.

8 Long Term Planning

The authoritative summary of the long-term planning system make be found in LDM-294. Here we expand upon the details of that system. The plan for the duration of construction is embodied in:

1. A series of *planning packages*, which describe major pieces of technical work. Planning packages are associated with concrete, albeit high-level, deliverables (in the shape of milestones, below), and have specific resource loads (staff assignments), start dates, and durations. The entire DM system is covered by around 100 of these planning packages.
2. *Milestones* represent the delivery or availability of specific functionality. Each planning package culminates in a milestone, and may contain other milestones describing intermediate results.

Planning packages are defined at the fourth level of the WBS breakdown (e.g. at 1.02C.04.02; see §5.1). They may not cut across the WBS structure, but rather must refer to that particular fourth-level element and its children.

Milestones are allocated to one of four levels, defined as follows:

Level 1 These are chosen by the NSF from a list defined by the Project.

Level 2 These reflect cross-subsystem commitments. As such, they must be defined in consultation with the DM Project Manager.

Level 3 These reflect cross-third-level WBS commitments. As such, they must be defined in discussion between two or more technical managers.

Level 4 These are internal to a particular third-level WBS, and can therefore be specified by a single technical manager.

Some of these are exposed to external reviewers: it is vital that these be delivered on time and to specification. Low-level milestones are defined for use within DM, but even here properly adhering to the plan is vital: your colleagues in other teams will use these milestones to align their schedules with yours, so they rely on you to be accurate.

Relationships may be defined between milestones and between milestones and planning packages. Often these are blocking relationships: a particular activity cannot proceed until all the work which blocks it has been completed. It is also possible to identify a non-specific relationship between activities. This should be taken to mean that they share some common aspects and hence it may be appropriate to consider them together.

8.1 Planning Research Work

In order for the DM system to reach its science goals, new algorithmic or engineering approaches must sometimes be researched. It is appropriate to budget time for this research work in planning packages. However, areas where successful delivery of the DM system is dependent on speculative research are a source of risk: wherever possible, the plan should also provide for a fallback option to be taken when research objectives are not achieved. When fallback options are not available, discuss how to account for this risk with the DM Project Manager (§3).

8.2 Earned Value and Planning Packages

A planning package has a duration and a staff assignment (it is "resource loaded"). Given a (nominal) cost per unit time of the staff involved (see §6.1), this translates directly to a BCWS.

During the cycle planning process (§9), effort is drawn from the budget embodied in the planning packages to generate the cycle plan, described in terms of epics: see §9.2.2 for details. Each epic itself has a particular budget. This budget is subtracted from that available in the planning package at the point when the epic is defined.

At any given time, the BCWP of a planning package consists of the sum of the BCWP of all epics derived from that package which have been marked complete, together with the fractions of value earned from all epics currently in progress.

An example may serve to illustrate.

Planning package P is baselined to start at the beginning of F17 and run through to the end of F18, i.e. a total of three cycles, or 18 months. It has two members of staff—A and B—assigned to it full time. Both share the same nominal cost of $\$X$ per cycle.

The BCWS for the total planning package is the cost per cycle multiplied by the number of cycles: $3 \times 2 \times \$X = \$6X$.

In F17, both members of staff are assigned to six-month epic derived from P. The BCWS of the epic is $\$2X$. The remaining value in the planning package is $\$4X$.

At the end of F17, the epic is completed. The BCWP and ACWP are both $\$2X$. The work is on cost and on schedule: there is no variance.

In S18, A is reassigned and is unable to work on a new epic derived from P. B continues the work alone, completing an epic worth $\$X$ by the end of the cycle. The BCWP and ACWP are now both $\$3X$; there is no cost variance. However, the BCWS is $\$4X$: compared to the original schedule for the planning package, there is a schedule variance of $-\$X$. There is a total of $\$3K$ left in the planning package.

In F18, C joins the project. C only costs $\$0.5X$ per cycle, but is a fast worker: she can complete in one cycle work that would take A or B two cycles.

B and C work together through F18. The ACWP for the cycle is $\$1.5X$; the BCWP is $\$3X$. The ACWP to date $\$4.5X$. The BCWP and BCWS are both $\$6X$. At this point, the project is complete: there is no schedule variance, and a cost variance of $+\$1.5X$.

8.3 Epic-Based Long Term Plans

As per §9, the epic is the standard level of granularity for planning work over the relatively short term (periods of several months). However, epics may also be valuable for longer-term, fine-grained planning. When a detailed description of work for a given planning package is known, it can and should be described in Jira through a series of epics assigned to the appropriate cycles. As long as they have not been scheduled for the current cycle, these epics can be freely created and changed at any time, without any sort of approval process. Of course, for this process to be practically useful, these epics should fit within the scope and budget of the relevant planning package.

Fine grained planning of this sort can be useful for "bottom-up" analysis of the work to be performed and validation of the resources needed to implement a particular planning package. Thinking through the plan in this way can help in building up a detailed plan in a flexible, agile way, while also ensuring that scope, cost and schedule are carefully controlled.

8.4 Software Releases

Per §9.1, a series of software releases will be made throughout LSST construction. These will provide a stable basis upon which external users (other subsystems, science collaborations and the wider community) can base their work.

In early development, releases followed a strictly time-based cadence. That is, they were made on a pre-defined schedule which tracked our short-term plan (§9), rather than being guaranteed to provide a particular set of functionality. As of F17 the test schedule is described in LDM-503, and releases will have to provide required functionality in support of those tests. Still there is no need for individual releases to be exposed as milestones above level 3 (exposing them at level 3 or below for internal use is optional). Other parts of the project which depend on certain functions being available should depend on a milestone describing that function, rather than on a particular release of the software.

In addition to this cyclical official release process, we shall provide packaged distributions of the codebase at more frequent intervals in support of commissioning or other activities that require a higher release cadence or timely delivery of particular features.

9 Short Term Planning

Short term planning is carried out in blocks referred to as cycles, which (usually) last for six months. Before the start of a cycle, technical managers work with the DM Project Manager, the relevant product owners and the Project Controls Specialist to ensure their plan for the cycle is well defined in both Jira and the PMCS.

9.1 Cycle Cadence & Release Planning

At the end of a cycle, a release manager appointed from within the Science Quality and Reliability Engineering (SQuaRE) team will coordinate a public release of the codebase. This release will consist of a coherent, well tested set of packages, together with release notes, documentation and performance characterization.

In order to make this possible, the release will be tagged two weeks before the end of the cycle. All work which is destined for the release must have been merged to the `master` branch by this point. For the remainder of the cycle, the priority is to provide bug fixes, documentation and other material in support of the release as requested by SQuaRE. In so far as it does not

interfere with that priority, other work may continue as normal, with the caveat that new development will not be included in a release until the end of the subsequent cycle.

Throughout this process, the SQuaRE technical manager will advertise the current state of the release to all interested parties using the LSST Community Forum.

Technical managers of the other groups are responsible for providing to SQuaRE such material as is required to support the release. This will include a set of release notes which provide a summary of work performed over the course of the cycle. Please liaise with SQuaRE in advance to establish the appropriate format and granularity of these notes.

9.2 Defining The Plan

9.2.1 Scoping Work

The first essential step of developing the short term plan is to produce an outline of the programme of work to be executed. In general, this should flow directly from the long term plan (§8), ensuring that the expected planning packages are being worked on and milestones being hit.

While developing the cycle, please:

- Do not add *artificial* padding or buffers to make the schedule look good;
- Do budget appropriate time for handling bugs and emergent issues;
- Reserve time for planning the following cycle: it will have to be defined before this cycle is complete;
- Leave time for other necessary activities, such as cross-team collaboration meetings and writing documentation.
- Per the cycle cadence (§9.1), ensure that new development will conclude (or, at a minimum, be in a releasable state) in time for the end of cycle release.

Obviously, ensure that the programme of work being developed is achievable by your team in the time available: ultimately, you will want to compare the number of SPs your team is able to deliver (§7) with the sum of the SPs in the epics you have scheduled (§9.2.2), while

also considering the skills and availability of your team. It is better to under-commit and over-deliver than vice-versa, but, ideally, aim to estimate accurately.

9.2.2 Defining Epics

The plan for a six month cycle fundamentally consists of a set of resource loaded epics defined in Jira. Each epic loaded into the plan must have:

- A concrete, well defined deliverable *or* be clearly described as a "bucket" (§9.2.4);
- The `cycle` field set to the appropriate cycle;
- The `wbs` field set to the appropriate WBS *leaf* element.
- The `Story Points` field set to a (non-zero!) estimate of the effort required to complete the epic in terms of SPs (see §7).

Be aware that:

- An epic may only be assigned to a single cycle. It is not possible to define an epic that crosses the cycle boundary (see §9.4 for the procedure when an epic is not complete by the end of the cycle).
- An epic may only be assigned to a single WBS leaf element. It is not possible to define epics that cover multiple WBS elements. See §9.2.6 for information on scheduling work which requires resources from multiple elements.
- An epic must descend from a single planning package (see §8).
- Although LOE work should be charged to the 00 fourth-level element (§6.3), this does not imply that other work cannot be charged here. Indeed, where possible management activities *should* be scheduled as epics with concrete deliverables in this element rather than being handled as LOE.
- The epic should be at an appropriate level of granularity. While short epics (a few SPs) may be suitable for some activities, in general epics will describe a few months of developer-time. Epics allocated multiple hundreds of story points are likely too broad to be accurately estimated.

The Project Controls Specialist (§3) will periodically (per §12) pull information from Jira to populate PMCS with the plan.

All epics which have WBS and cycle defined will be loaded into PMCS (and must, therefore, have concrete deliverables and plausible SP estimates). Epics which do not satisfy these criteria may be defined in Jira. These will not be pulled into PMCS, will not form part of the scheduled plan, and will not earn value. However, they may still be useful for organizing other work, sketching plans for future cycles, etc: please define them as necessary.

In order to fully describe the plan to PMCS, epics require information that is not captured in Jira. Specifically, it is necessary to define:

- Start and end dates for the epic;
- Staff assignments.

Although it is possible—indeed, encouraged—to set the `assignee` field in Jira to the individual who is expected to carry out the bulk of the work in an epic, this does not provide sufficient granularity for those cases when more than one person will be contributing.

In fact, it is only required to provide a staff assignment in terms of "resource types" (i.e. scientists, senior scientists, developers, senior developers, etc). In practice, to ensure your team is evenly loaded, it is usually necessary to break it down to named individuals.

This information is most conveniently captured in per-team spreadsheets which are supplied to the Project Controls Specialist before the start of the cycle. Spreadsheets describing previous cycles are stored in Google Drive: a convenient way to get started would be to use one of those as a template.

The spreadsheets used capture epic start and end dates at monthly granularity. This can lead to a variance (see §6) when monthly results are tabulated (it assumes that work for an epic is evenly distributed across all the months in which it is scheduled). In practice, this variance is likely to be small, and should average out by the end of the cycle, when all epics should be closed on schedule. However, if this becomes a problem, it is possible to fine-tune dates by directly consulting with the Project Controls Specialist.

When loading epics at the start of a cycle, it is not necessary that they be fully loaded with stories (defined as per §10.1): these can be defined during the cycle. You do, of course, need

to have thought through the contents of the epic in enough detail to provide an overall SP estimate and deliverables, though.

With the agreement of the Project Manager and Project Controls Specialist, it is acceptable to load the plan for a cycle in three month "chunks". That is, the plan for the first three months of the cycle is loaded before the start of the cycle, and the remaining part of the plan covering the final three months is loaded before the start of the fourth month. This approach provides an opportunity to fine-tune the plan for the second half of the cycle, without requiring a formal LSST Change Request (LCR) (§9.3).

9.2.3 Scheduling Research Work

As discussed in §8.1, research is sometimes required to meet our objectives. However, it is not a natural fit to our usual planning process, as it is speculative in its nature: it is often impossible to produce a series of logical steps that will lead to the required result. We acknowledge, therefore, that scheduling an epic to deliver some particular new algorithm based on the results of research is impossible: we cannot predict with any confidence when the breakthrough will occur.

We therefore schedule research in timeboxed epics: we allocate a certain amount of time based on the resources available, rather than on an estimate of time to completion. However, note that these timeboxed epics should still provide concrete deliverables: they are not open-ended "buckets" as discussed elsewhere. Since we cannot rely on the successful completion of the research project as a deliverable, we instead require that a summary of the research completed to date be delivered at the completion of the time allocated. The presentation and format of this report will vary depending on the nature of the research (a technical note is a likely option), and, as usual (§9.2.2), should be defined before the epic is ingested to PMCS.

9.2.4 Bucket Epics

Some work is "emergent": we can predict in advance that it will be necessary, but we cannot predict exactly what form it will take. The typical example of this is fixing bugs: we can reasonably assume that bugs will be discovered in the codebase and will need to be addressed, but we cannot predict in advance what those bugs will be.

This can be included in the schedule by defining a "bucket" epic in which stories can be created when necessary during the course of a cycle. Make clear in the description of the epic that this

is its intended purpose: every epic should either have a concrete deliverable or be a bucket.

Bucket epics have some similarities with LOE work. As such, we acknowledge that they are necessary, but seek to minimize the fraction of our resources assigned to them. If more than a relatively small fraction of the work for a cycle is assigned to bucket epics, please consider whether this is really necessary and appropriate.

Be aware that even bucket epics must be assigned to a specific *leaf* element of the WBS. That is, it is not in general possible to define an epic which handles bug reports or emergent feature requests across the whole of the codebase unless a specific WBS leaf element is devoted to maintenance activities of this type. Instead, it may be necessary to define a different bucket epic for each leaf of the WBS tree.

9.2.5 Mapping SPs to BCWS

As discussed above, the amount of work to be performed is estimated in terms of SPs (§7), while the earned value (§6) system accounts for work in terms of budgeted cost (BCWS). In order to estimate the value earned by completing an epic, it is necessary to map from one to the other.

The outline of the calculation here is straightforward: SPs map to developer hours. Given the staff assignment for the epic (see §9.2.2), the number of hours scheduled per developer can be calculated. Given the nominal costs (per §6.1) associated with each developer, the total labor cost can be estimated.

Therefore, we calculate the number of hours of each staffing grade being assigned to the epic, multiply that by the cost per hour of that grade, and that provides the cost of the work scheduled.

9.2.6 Cross Team Work

Planning epics are always assigned to a particular WBS leaf element: they do not span elements or teams. It is therefore impossible to schedule a single epic which covers cross-team work. There are two ways to approach this problem:

- The technical managers for both teams to be involved in the work schedule epics separately, within their own WBS structure. They are responsible for agreeing start and end

dates, deliverables and resourcing between themselves. From the point of view of the PMCS, these epics are independent pieces of work which happen to be coincident.

- With agreement between technical managers, an individual may be detached from one team and explicitly work for another team for some defined period. One technical manager is therefore responsible for defining and scheduling their work. Their "home" manager will charge actuals (§11 against the WBS supplied by the manager manager of the receiving team.

Regardless of the approach taken, technical managers should be especially careful to ensure that cross-team work is well defined. Usually, it is convenient for a single manager to take ultimate responsibility for ensuring that it is successfully delivered.

9.3 Revising the Plan

During the cycle, it is possible that changing circumstances will cause reality not to exactly match with the plan. This will ultimately cause a variance (see §6), which should be minimized and which—if it becomes significant enough—will require a narrative.

After the plan for the cycle has been entered into Jira, it is under change control: it can only be altered through a LCR approved by the Change Control Board (CCB). In order to reschedule (or remove entirely from the cycle) an epic which has not yet started, the technical manager should work with the Project Controls Specialist (§3) to prepare and submit an appropriate LCR to the CCB. The CCB meets on the third Wednesday of the calendar month; change requests must be submitted well in advance of this. Therefore, it is advisable to take time early in the calendar month to review epics due to start in the *following* month and to issue an LCR on them if necessary.

Note that it is *not possible* to alter history by means of an LCR. That is, if the scheduled start date of an epic is already in the past, it is not possible to move it into the future using a change request. In this case, there is no option but to carry the variance related to the late start of the epic into the future, to describe that with narratives (§6.2) where necessary, and to attempt to address the variance as soon as is possible.

Based on the above, it is clear that technical managers should closely track performance relative to the plan throughout the cycle, and proactively file change requests to avoid running variances wherever possible.

9.4 Closing the Cycle

Assuming everything has gone to plan, by the end of a cycle all deliverables should be verified and the corresponding epics should be marked as done. Marking an epic as done asserts that the concrete deliverable associated with the epic has been provided. The total cost of that functionality—the BCWS, calculated as per §9.2.5—is now claimed as value earned.

Epics which are in progress at the end of the cycle cannot be closed until they have been completed. These epics will spill over into the subsequent cycle. It is *not* appropriate to close an in-progress epic with a concrete deliverable until that deliverable has been achieved: instead, a variance will be shown until the epic can be closed. Obviously, this will impact the labor available for other activities in the next cycle. (This does not apply to bucket epics (§9.2.4), which are, by their nature, timeboxed within the cycle).

Similar logic applies to epics which *have not been started*: if the planned start date is in the past, they can no longer be rescheduled by means of an LCR (§9.3). They must be completed at the earliest possible opportunity; you will show a variance until this has been done.

10 Execution

Having defined defined the plan for a cycle following §9, we execute it by means of a series of month-long sprints. In this section, we detail the procedures teams are expected to follow during the cycle.

10.1 Defining Stories

Epics have already been defined as part of the cycle plan (see §9.2.2). However, the epic is not at an appropriate level for scheduling day-to-day work. Rather, each epic is broken down into a series of self-contained “stories”. A story describes a planned activity worth between a small fraction of a SP and several SPs (more than about 10 is likely an indication that the story has not been sufficiently refined). It must be possible to schedule a story within a single sprint, so no story should ever be allocated more than 26 SPs.

The process for breaking epics down into stories is not mandated. In some circumstances, it may be appropriate for the technical manager to provide a breakdown; in others, they may request input from the developer who is actually going to be doing the work, or even hold a brainstorming session involving the wider team. This is a management decision.

It is not required to break all epics down into stories before the cycle begins: it may be more appropriate to first schedule a few exploratory stories and use them to inform the development of the rest of the epic. However, do break epics down to describe the stories which will be worked in an upcoming sprint (§10.2) before the sprint starts. When doing so, you may wish to leave some spare time to handle emergent work (discussed in §10.4).

Note that there is no relationship enforced between the SP total estimated for the epic and the sum of the SPs of its constituent stories. It is therefore possible to over- or under-load an epic. This will have obvious ramifications for the schedule. See §10.5 for a discussion of its impact on earned value.

10.2 Sprinting

Each team organizes its work around periods of work called sprints. A sprint comprises a defined collection of stories which will be addressed over the course of the month. These stories are not necessarily (indeed, not generally) all drawn from the same epic: rather, while epics divide the cycle along logical grounds, sprints divide it along the time axes.

Broadly, executing a sprint falls into three stages:

1. Preparation.

   The team assigns the work that will be addressed during the sprint by choosing from the pre-defined stories (§10.1). Each team member should be assigned a plausible amount of work, based on the per-story SP estimates and the likely working rate of the developer (see §7).

   The process by which work is assigned to team members is a local management decision: the orthodox approach is to call a team-wide meeting and discuss it, but other approaches are possible (one-to-one interactions between developers and technical manager, managerial fiat, etc).

   Do not overload developers. Take vacations and holidays into account. The sprint should describe a plausible amount of work for the time available.

2. Execution.

   Daily management during the sprint is a local decision. Suggested best practice includes holding regular “standup” meetings, at which developers discuss their current activities and try to resolve “blockers” which are preventing them from making progress.

Stories should be executed following the instructions in the Developer Guide as regards workflow, coding standards, review requirements, and so on. It is important to ensure that completed stories are marked as done: experience suggests that this can easily be forgotten as developers rush on to the next challenge, but it is required to enable us to properly track earned value as per §10.5.

When completing a story we do not change the number of SPs assigned to it: the SP total reflects our initial estimate of the work involved, not the total time invested. However, we should *also* record the true SPs expended on the issue. This makes it possible to review the quality of our estimates at the end of the sprint. Each individual, with guidance from their T/CAM, should use this information as they strive to improve the accuracy of their planning and estimating.

Avoid adding more stories to a sprint in progress unless it is unavoidable (for example, the story describes a critical bug that must be addressed before proceeding). A sprint should always stay current and should be up-to-date with reality; if necessary, already scheduled stories may be pushed out of a sprint as soon as it is obvious it is unrealistic to expect them to be completed.

3. Review.

   At the end of the sprint, step back and consider what has been achieved. What worked well? What did not? How can these problems be avoided for next time? Was your estimate of the amount of work that could be finished in the sprint accurate? If not, how can it be improved in future? Refer to the burn-down chart for the sprint, and, if it diverged from the ideal, understand why.

   Again, the form the review takes is a local management decision: it may involve all team members, or just a few.

We use Jira's Agile capabilities to manage our sprints. Each technical manager is responsible for defining and maintaining their own agile board. The board may be configured for either Scrum or Kanban style work as appropriate: the former is suitable for planned development activities (e.g. Science Pipelines development); the latter for servicing user requests (e.g. providing developer support).

10.3 Completing Epics

An epic may be marked as done when:

1. It contains at least one completed story;
2. There are no more incomplete stories defined within it;
3. There are no plans to add more stories;
4. (If applicable, i.e. it is not a bucket, as defined in §9.2.4) its concrete deliverable has been achieved.

Note that it is not permitted to close an epic without defining at least one story within it. Empty epics can never be completed.

When an epic is marked as complete, all of its value is earned (§10.5).

10.4 Handling Bugs & Emergent Work

10.4.1 Receiving Bug Reports

Members of the project who have access to Jira may report bugs or make feature requests directly using Jira. As discussed in §10.6, technical managers should regularly monitor Jira for relevant tickets and ensure they are handled appropriately.

Our code repositories are exposed to the world in general through GitHub. Each repository on GitHub has a bug tracker associated with it. Members of the public may report issues or make requests on the GitHub trackers. Per the Developer Workflow, all new work must be associated with a Jira ticket number before it can be committed to the repository. It is therefore the responsibility of technical managers to file a Jira ticket corresponding to the GitHub ticket, to keep them synchronized with relevant information, and to ensure that the GitHub ticket is closed when the issue is resolved in Jira.

The GitHub issue trackers are, in some sense, not a core part of our workflow, but they are fundamental to community expectations of how they can interact with the project. Ensure that issues reported on GitHub are serviced promptly.

In some cases, the technical manager responsible for a given repository is obvious, and they can be expected to take the lead on handling tickets. Often, this is not the case: repositories regularly span team boundaries. Work together to ensure that all tickets are handled.

10.4.2 Issue Types

We have previously referred to day-to-day work being described by means of stories. However, Jira provides us with two additional issue types: "bug" and "improvement". Per RFC-43, the semantics of the various issue types are:

- A story is the result of breaking down an epic into workable units;
- A bug describes a fault or error in code which has already been accepted to master;
- An improvement describes a feature request or enhancement which has not been derived by breaking down the long term plan (i.e., it is an ad-hoc developer or user request).

The three issue types are functionally equivalent: these semantic distinctions are for convenience only, and are not rigorously enforced.

In particular, note that all issue types are equivalent in terms of the data which is loaded to the PMCS: it makes no distinction between them. Marking a bug or improvement as `done` has exactly the same impact on the global earned value state as would completing an equivalent story.

10.4.3 Scheduling

In some cases, a ticket may describe emergent work which must be addressed immediately by adding it to a bucket epic (§9.2.4). In other cases, it can be deferred to a later cycle, or, after appropriate discussion, may be regarded as inappropriate (and can be tagged as `invalid` or `won't fix`). This is a management decision. When closing a ticket as inappropriate, please take a moment to describe why—the individual who reported it will appreciate an explanation of why it has been rejected, and it will serve as a useful reference the next time somebody suggests the same thing.

A special case of inappropriate tickets are those that duplicate work which has already been described elsewhere. Please close these as `invalid`, and add a Jira link of type `duplicates` to the original ticket.

Tickets which are obviously filed by mistake may simply be deleted rather than setting a special status. Please only do this when you are sure there is no value to leaving an audit trail,

and when you have verified that the original author of the ticket is aware of and understands the outcome.

10.4.4 Relationship to Earned Value

We adopt the position that bugs are a natural part of the software lifecycle, and hence addressing them at an appropriate level earns value in the same way as new software development. That is, SPs earned by working on bugs and completing bucket epics contribute to earned value in the same way as other work.

However, bugs do serve as an bellwether for software quality issues. It would obviously be inappropriate—and a severe source of schedule risk—for the value earned from addressing bugs in existing software to dominate the productivity of the team at the expense of new development. We expect that no more than around 30% of schedulable developer time will be dedicated addressing bugs and performing maintenance: any more than this must be carefully justified.

10.5 Earning Value

The basic procedure for earning value during the cycle is akin to that discussed in §8.2 for long term planning.

In short, as we saw in §9.2.5, the BCWS for a particular epic is defined by its *estimated* (i.e. attached to the epic before work commences) SP total and its staff assignment. When an epic is marked as complete (following the criteria in §10.3, this is the value that is earned.

The BCWP for an epic is calculated based on the fractional completeness of an epic. That is, if an epic has a total SP count of X, and the total of stories marked as complete within it is Y, then $\mathrm{BCWP} = \mathrm{BCWS} \times Y/X$.

Be aware that stories that marked as `invalid` or `won't fix` in Jira are not included in this calculation: they earn no value.

As we saw in §10.1, it is not required that the total SPs of all the stories contained within an epic (the “planned SPs”) is equal to the total SP estimate of the epic itself (“estimated SPs”). Further, it is permitted to add stories to (or, indeed, remove stories from) the epic during the cycle. In these cases, we hold to two basic tenets:

1. No epic can ever be more than 100% complete;
2. Completeness cannot decrease. That is, if an epic has been registered as 90% complete, adding more stories cannot make it *less* complete than before.

In order to meet these criteria, the relative weights of stories will be automatically adjusted on ingest to the PMCS. The detailed algorithm by which this adjustment is made is not publicly documented.

10.6 Jira Maintenance

At any time, new tickets may be added to Jira by team members. Please remind your team of the best practice in this respect (RFC-147). It is the responsibility of technical managers to ensure that new tickets are handled appropriately, updating the schedule to include them where necessary.

It is required that the `Team` field be set to the appropriate team (RFC-145). This indicates which manager is responsible for seeing that the work is completed successfully. Available teams, and the associated managers, are listed in the Developer Guide; generally speaking, they align with the the work breakdown structure described in §5.1. Where there is uncertainty about which team should be responsible for a particular ticket, the "System Management" team may be used to indicate that the DM Project Manager is responsible for assigning the work.

Please regularly monitor Jira for incomplete tickets and update them appropriately. Where tickets describe bugs or other urgent emergent work which cannot be deferred, refer to §10.4.

10.7 Coordination Standup

The technical managers meet with the Project Manager (§3) and interested others—it is not a closed meeting—twice every week. This is a forum to discuss general project management issues, but, in particular, to resolve issues which cut across team boundaries and are relevant for ongoing sprints.

Meetings take take place online, using a pre-arranged URL[2], at 11 a.m. Project (Pacific) Time on Tuesdays and Fridays.

[2]The meeting URL is not included here; contact the Project Manager for details.

10.8 Monthly Progress Narratives

Every calendar month, each technical manager is required to support the Project Manager with a report on the activities of their group. This report should be generally submitted no later than tenth of the month (refer to §12), but this may be moved earlier on occasion. You are encouraged to submit your report as early in the month as possible.

Submit your report by editing the template for the appropriate month on Google Docs[3]. You need to fill in all the sections with your name attached; when complete, remove your name. Provide a brief (one or two sentences) high level summary, a per-WBS breakdown of work over the month being reported on and plans for the upcoming month, as well as describing any recruitment activities (positions opened, interviews conducted, appointments made, etc). Refer to previous reports for examples of the style used (but note that they are not not always consistent).

11 Reporting Actuals

In order to comply with the earned value management system (§6), it is necessary to track the actual cost of work being performed (the "actuals") in each leaf element of the WBS. That is, whenever an invoice is issued from a subcontracting institution to AURA, it must be broken down into dollar charges against individual WBS elements.

Some institutions rigorously track how staff are spending their time (e.g. by filling in timesheets), and may directly make that information available to AURA as part of the invoicing process. In this case, the technical manager need take no further action.

Other institutions do not rigorously check staff activity and/or do not supply this information to AURA when invoicing. In this case, the technical manager is responsible for breaking down the invoice by WBS and forwarding that to the relevant AURA contracts officer (check with the Project Manager (§3) if you are unsure who that is). Note that, since SPs reflect estimated, not actual, time spent on work (§10.2), it is *not* appropriate to simply allocate actual costs based on SP totals.

Typically, expenses are accrued at a broadly constant rate for each individual: salaries do not vary much from month to month. However, in some months, a given developer may be significantly less productive than others (for example, due to paid vacation). In these cases, it

[3] Reports are linked from the Useful DMLT Links Confluence page.

Invoice Date: YYYY-MM-DD			Period: YYYY-MM-DD/DD		
Total	02C.0N.00	02C.0N.01	02C.0N.02	...	02C.0N.0M
	KLM20N00A	KLM20N01A	KLM20N02A	...	KLM20N0MA
$ABCD.EF	$GHIJ.KL	$MNOP.QR	$STUV.WX	...	$YZ.00

TABLE 6: Example invoice breakout showing dollar values allocated to both WBS elements and corresponding account numbers.

is appropriate to spread the cost across all the WBS elements the developer has been working on.

A typical invoice breakout should be supplied in a spreadsheet similar to that shown in Table 6.

Note that when reporting actuals at this level it is not required to provide a mapping from dollar values to individuals who did the work. However, it is important to note that, should the Project be audited in the future, it is perfectly possible that they will wish to examine such a mapping. You should therefore keep records which will enable you to provide it upon request.

12 Standard Reporting Cycle

- During the first week of the calendar month, data from Jira together with actual costs (labor charges, etc) are ingested to the PMCS system. This indicates the progress of all activities and shows any Earned Value variances. This information is made available to technical managers through eCAM.
- During the second week of the calendar month:
  - Variance narratives (§6.2), where necessary, must be submitted through eCAM.
  - The monthly progress narrative (§10.8) must be submitted through Google Docs by the tenth day of the month.
- The DM Project Manager assembles extended and summary reports, based on the reports received from the institutions. The extended report is periodically examined by Federal auditors, while the summary report is provided to senior management and the AMCL for review.

13 Personnel

13.1 Staffing Changes

In addition to onboarding procedures at your local institution, please be aware of

- The LSST New Employee Onboarding material, and
- The DM Developer Onboarding Checklist

and direct new recruits to them when they join your team[4].

The responsible T/CAM must also complete an onboarding form for the new recruit and update the DM Team spreadsheet. Similarly, when members of staff team leave the project, the T/CAM should fill in an offboarding form.

13.2 Collaborators

DM and LSST collaborate with many outside individuals. We also use many tools such as Jira, Slack and Confluence which are licensed for a certain number of users. To foster collaboration it is useful that some close collaborators who are not part of the project have accounts on such systems. Requests for such accounts should be made via Jira tickets in the "IHS" project, with explicit review requested from the DM Project Manager and Deputy Project Manager. Collaborators granted such accounts should make use of them: if they are not actively used for six months the accounts shall be revoked.

14 Glossary

ACWP Actual Cost of Work Performed (often referred to as "actuals").
BCWP Budgeted Cost of Work Performed.
BCWS Budgeted Cost of Work Scheduled.
budgeted (labor) unit An hour of work.
CAM Control Account Manager. A CAM is responsible for the scope, schedule and budget for one or more control accounts.
CCB Change Control Board. All changes to the baselined plan must be approved by the CCB. See LPM-19 for details.

[4] As per §7.2.1, remember that newcomers should be allocated SPs for working through this material.

control account An intersection point between the WBS and the OBS. For example, work performed at IPAC (1.03) on the Science User Interface (1.02C.05) is managed by a single control account.

CPI Cost Performance Index. Defined as BCWP ÷ ACWP.

CV Cost Variance. Defined as BCWP − ACWP.

cycle The time period over which detailed, short-term plans are defined and executed. Normally, cycles run for six months, and culminate in a new release of the LSST Software Stack, however this need not always be the case.

eCAM The eCAM Notebook, a tool which reports information from the PMCS. It provides a convenient view of the current status of the project in terms of Earned Value Management System (EVMS).

element A node in the hierarchical project WBS.

epic A self contained work with a concrete deliverable which my be scheduled to take place with a single cycle and WBS element.

EVMS Earned Value Management System. See the brief description in §6, or refer to formal training.

Jira Issue and project tracking software produced by Atlassian. LSST's Jira is a core interface between technical managers, their teams, and the PMCS.

LCR LSST Change Request. It is necessary to submit a change request to alter any "baselined" aspect of the project. This includes, for example, altering change controlled plans, or epics that have been loaded to the PMCS.

LOE Level of Effort. LOE work is that which does not correspond to a specific deliverable. A detailed definition is provided in LDM-472; see also the discussion in §6.3.

MREFC Major Research Equipment and Facilities Construction. The terms under which LSST's NSF funding has been issued; we are required to strictly adhere to them.

NSF National Science Foundation.

OBS Organizational Breakdown Structure.

PMCS Project Management Control System. The PMCS is not a single piece of software, but rather an interlocking suite of tools. In general, the CAM need not interact with PMCS directly, but only through the eCAM and Jira tools: it is safe to treat PMCS as a "black box". Occasionally, individual PMCS components such as Primavera or Deltek Cobra escape this abstraction and appear in documentation.

resource loading Assigning particular resources (in software development, almost always staffing) to particular tasks. A "resource loaded plan" provides a mapping of resources to the plan throughout execution.

risk Risks are (per ISO 31000) "the effect of uncertainty upon objectives". For the purposes of

this document, that corresponds to the impact of unplanned or unpredictable events upon the cost or schedule of the Project. The Project maintains a register of risks, which includes probability estimates and possible mitigations.

SP Story Point. Used to estimate the duration of tasks in Jira. One SP is equivalent to 4 hours of uninterrupted effort by a competent developer.

SPI Schedule Performance Index. Defined as BCWP ÷ BCWS.

sprint A defined period of work for a particular team. Typically, sprints are one calendar month long, but this is not required.

SQuaRE Science Quality and Reliability Engineering. One of the teams which makes up the Data Management Group. SQuaRE coordinates the end-of-cycle release of the codebase (refer to §9.1), and therefore plays a pivotal role in planning.

story A Jira issue type describing a scheduled, self-contained task worked as part of an epic. Typically, stories are appropriate for work worth between a fraction of a SP and 10 SPs; beyond that, the work is insufficiently fine-grained to schedule as a story. While fractional SPs are fine, all stories involve work, so the SPs total of an in progress or completed story should not be 0.

SV Schedule Variance. Defined as BCWP − BCWS.

timebox A limited time period assigned to a piece of work or other activity. Useful in scheduling work which is not otherwise easily limited in scope, for example research projects or servicing user requests.

WBS Work Breakdown Structure.

15 References

[1] **[LPM-19]**, Angeli, G., McKercher, R., 2015, *Change Control Process*, LPM-19, URL https://ls.st/LPM-19

[2] **[LDM-472]**, Becla, J., Economou, F., Mueller, F., et al., 2017, *LSST DM Project Management and Tools*, LDM-472, URL https://ls.st/LDM-472

[3] **[LPM-81]**, Kantor, J., Krabbendam, V., 2015, *Cost Estimating Plan*, LPM-81, URL https://ls.st/LPM-81

[4] **[LPM-98]**, Long, K.E., 2016, *LSST Project Controls System Description*, LPM-98, URL https://ls.st/LPM-98

[5] **[LPM-43]**, McKercher, R., 2016, *WBS Structure*, LPM-43, URL https://ls.st/LPM-43

[6] **[LPM-44]**, McKercher, R., 2016, *WBS Dictionary*, LPM-44, URL https://ls.st/LPM-44

[7] **[LDM-503]**, O'Mullane, W., Jurić, M., Economou, F., 2017, *Data Management Test Plan*, LDM-503, URL https://ls.st/LDM-503

[8] **[LDM-294]**, O'Mullane, W., Swinbank, J., Jurić, M., DMLT, 2017, *Data Management Organization and Management*, LDM-294, URL https://ls.st/LDM-294