Rubin Observatory

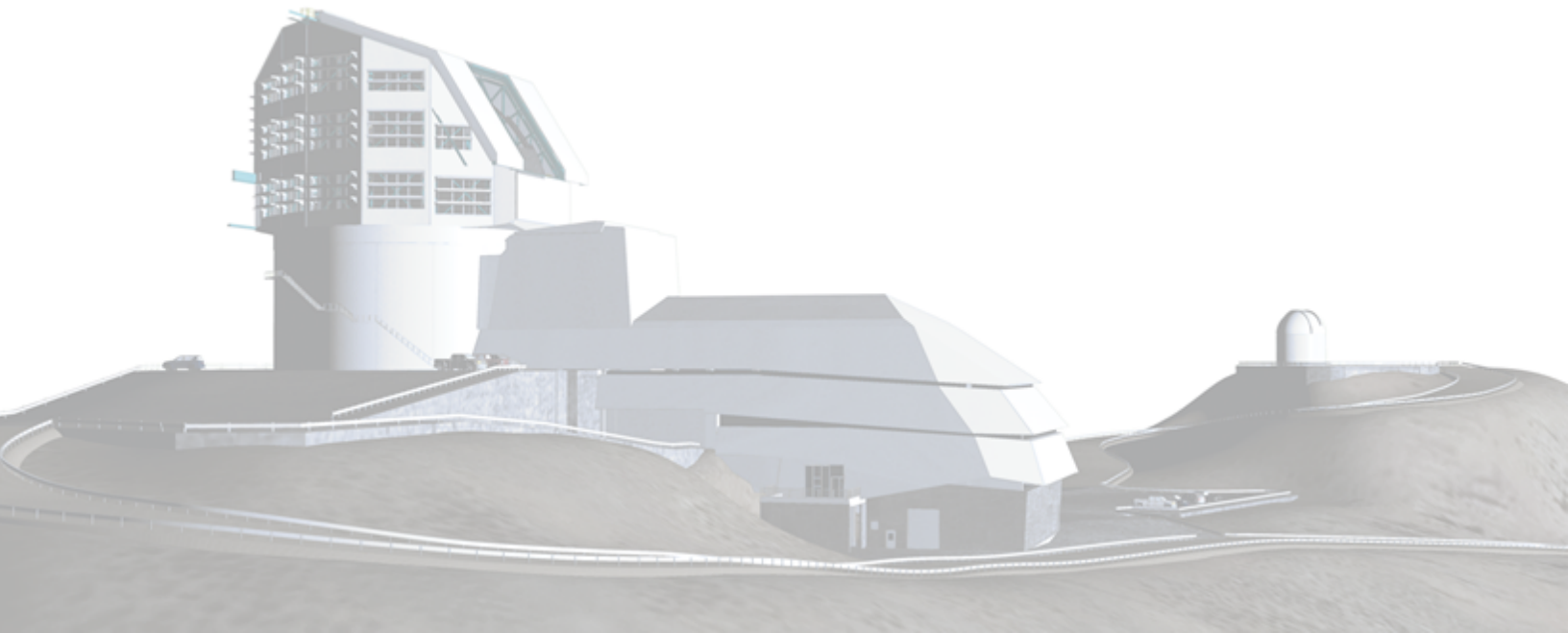
Vera C. Rubin Observatory
Data Management

Management principles and use of Jira for Rubin Operations

William O'Mullane and Amanda Bauer

RTN-005

Latest Revision: 2020-08-27



Rubin Observatory

Abstract

This document describes guidelines for the management of effort in Vera C Rubin Operations. It describes the process for agile-based work, including tasks that are carried out with regularity (nightly or monthly, etc), long-term development work that iteratively incorporates user feedback, and Level of Effort work as well.

Rubin Observatory

Change Record

Version	Date	Description	Owner name
1	YYYY-MM-DD	Unreleased.	William O'Mullane

Document source location: `https://github.com/rubin-observatory/rtn-005`

Rubin Observatory

Contents

Rubin Observatory

Management principles and use of Jira for Rubin Operations

1 Introduction

This document provides a guide to the Vera C. Rubin Observatoryapproach to project management. See also the the operations proposal LDO-31 In operations, there is no Primavera or Earned Value Management System (EVMS) as there are for the Rubin Construction Project.

There are a few main ways work will be managed, all within an Agile framework. The Agile method is a set of practices that allow the ability to adapt and respond to change through collaboration between self-managing, cross-functional teams. We focus on the people doing the work and how they work together, instead of organizing the program into functional silos.

This framework allows the multidisciplinary Rubin team to operate the facility and generate nightly data products while continuously improving efficiency of workflows ('kanban' style), as well as iteratively responding to user feedback on a longer timescale to maximize the scientific benefit of annual data releases ('scrum' style).

Some examples that will use the kanban style are the daily functionality checks needed for nightly operations of the telescope or the regular monitoring of progress toward achieving milestones and reporting to governing organizations. For either of these examples, we will maintain a set of tasks that need to be performed in sequence every cycle. Each task could be described individually inside a JIRA ticket and grouped together in a set, so that any person doing the daily checks, for example, could be assigned the set of tasks, open the tickets, comment on results, pass additional work onto others as needed, and mark the tickets done as the work is completed. In this way, the completion of the checklist of tasks is transparent and traceable and can be monitored, and the efficiency of the steps in the flow of work can be improved upon continuously.

Examples of scrum-like work include improving algorithms in response to user community feedback, optimizing the observing strategy as the survey progresses, and other incremental work needed to produce the annual data releases.

In this document, we lay out the procedural details for how we define and carry out plans,

following the framework described above, to deliver on our milestones, maintain visibility in our workflows, remain responsive to change, and offer staff the ability to innovate and collaborate.

2 Useful Contacts

The Rubin ObservatoryActing Director is Robert Blum, the Interim Deputy Director for NOIRLab is Amanda Bauer, and the Deputy Director for SLAC is Phil Marshall. They are the first point of contact for all issues regarding project management within Rubin Observatory.

The Project Controls Specialist is Cathy Petry. She monitors the budgets and maintains details within WeBud, the NOIRLab budgeting system. She assists in tracking milestones and reporting.

3 Associate Directors

Rubin Operations has four operational Departments and the Director's Office: RDO (Rubin Director's Office), ROO (Rubin Observatory Operations), RDP (Rubin Data Production), RSP (Rubin System Performance), and REO (Rubin Education and Public Outreach). Each operational Department is lead by an Associate Director.

4 Formal Organizational Structure

4.1 Work Breakdown Structure

The Work Breakdown Structure (WBS) provides a hierarchical index of all hardware, software, services, and other deliverables which are required to operate Rubin Observatory. Thus:

WBS	Description	Lead
NUM	DESC	Lead

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”.

Nodes in the WBS tree are referred to as elements. All of these third level WBS elementss (activities) are subdivided, forming a fourth level. The fourth level elements’ numbers 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.

4.2 Organization Breakdown Structure

4.3 The Control Account Manager

In general, a Control (or Cost) Account (CA) is the intersection between the WBS and the Organisation Breakdown Structure (OBS). Each CA falls under the purview of a CAMera (CAM). At Rubin Observatory, a single Team Leader is responsible for the whole of a third level WBS, and therefore acts as the CAM. That is, the Team Leader is the manager for that particular WBS element, and is responsible for all work performed on it, even though the work may be performed by staff outside the Team Leader’s institution.

4.4 Level of Effort Work

There is work throughout Rubin Operations that will be recorded as a LOE. These activities include attending meetings, reporting on milestones or taking 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. With LOE work is assumed to earn value (informally) 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.

As an example, our first-order estimate is that developers in Data Production 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: Team Leaders 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)

5 Estimating Effort

5.1 Basic Assumptions

The Construction 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. The Rubin operations partners, SLAC and NOIRLab, may have different definitions for tracking their staff time; Rubin operations uses 1800 hours per year as a fiducial value for effort estimation purposes.

In general, staff in Rubin operations roles at a given expected full-time equivalent (FTE) effort level are expected to devote that fraction of their total work time to Rubin Observatory.

Staff in "scientist" roles are expected to spend 20% of their time on personal research (see the Rubin Operations Plan for details). That is, scientists are expected to devote 1,440 hours per year to operations activity, and the remainder of their time to personal research.

Personal research time is charged to Rubin along with the operations role time, the logic being that scientists in operations roles must be engaged in research as well in order to succeed in their operations role.

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

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

$Byte(8bit)(B)/1440$. A scientist with total annual cost B ALSO charges an hourly rate of $B/180$.

No further corrections are necessary. In particular, there is no difference in the way working hours are measured, or the conversion of Survey Performances (SPs) to hours. (See below for more discussion of story points.)

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

TABLE 3: Expected working rates for developers and scientists. Technicians and engineers follow the same rates as developers.

In Data Production, our base assumption is that 30% of an individual's Rubin Observatory-operations time (i.e. 540 hours/year for a full-time developer, 432 hours/year for a full-time 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 §4.4). 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 Story Points (SP), with one SP being defined as equivalent to four hours of effort (half a day's work) by a competent developer. Thus, we expect developers and scientists to produce 26.25 and 21 SPs per *average* month respectively.

5.2 Special Cases

5.2.1 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

[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.

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 (§4.1).

5.2.2 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.

6 Long Term Planning

The authoritative summary of the long-term planning system may be found in … Here we expand upon the details of that system. The plan for pre-operations and Survey Operations is embodied in:

1. A set of *Milestones*, each of which represents the delivery or availability of specific functionality. Each planning package culminates in a milestone, and may contain other milestones describing intermediate results.
2. A series of *epics*, which describe major pieces of technical work. Epics are associated with concrete, albeit high-level, deliverables in the shape of the above milestones, and have specific resource loads (staff assignments), start dates, and durations.

Milestones are allocated to one of four levels, defined as follows:[2]

Level 1 These are at the full observatory level owned by the directors office

Level 2 These reflect cross department commitments. As such, they must be defined in consultation with the Director's Office

[2]Note that the level of the milestone is logically distinct from the level of the WBS and OBS, even though the same term ("level") is used to label both. In the OBS, level 0 is the Observatory, level 1 is the departments, level 2 is the teams, and level 3 is the groups within the teams. We take care to always disambiguate "WBS level" from "milestone level" and never use the term "level" without these qualifiers preceding it.

Level 3 These are internal to a particular Department and assigned to a team and can therefore be specified by a single team lead.

Some of these milestones are exposed to external reviewers: it is vital that these be delivered on time and to specification. Low-level (3 and beyond) milestones are defined for use within Departments, 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.

Epics should help achieve milestones i.e. they may be blocking issues on the milestones. A detailed description of work for a given epic is known, it can and should be described in JIRA. This should be assigned to the appropriate cycles.

6.1 Planning Research Work

In order for Rubin Observatoryto 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.

6.2 Epic-Based Long Term Plans

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.

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.

7 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, milestones should be confirmed by the Director's Office.

7.1 Defining The Plan

7.1.1 Scoping Work

The first essential step of developing the short term plan is to produce an outline of the program of work to be executed. In general, this should flow directly from the long term plan (§6), 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 (§**??**), 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 (§5) with the sum of the SPs in the epics you have scheduled (§7.1.2), 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.

7.1.2 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” (§7.1.4);

- 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 §5).

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 §7.2 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 §**??** for information on scheduling work which requires resources from multiple elements.
- An epic must descend from a single planning package (see §6).
- 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.

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.

7.1.3 Scheduling Research Work

As discussed in §6.1, 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.

7.1.4 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.

7.2 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.

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 (§7.1.4), which are, by their nature, timeboxed within the cycle).

8 Execution

Having defined defined the plan for a cycle following §7, 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.

8.1 Defining Stories

Epics have already been defined as part of the cycle plan (see §7.1.2). 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 (§8.2) before the sprint starts. When doing so, you may wish to leave some spare time to handle emergent work (discussed in §8.4).

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.

8.2 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 (§8.1). 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 §5).

   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 §**??**.

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 Technical/Control (or Cost) Account Manager (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).

8.3 Closing Epics

8.3.1 Completing the Work

An epic may be marked as done when:

1. It contains at least one completed story;
2. There are no more incomplete storys defined within it;
3. There are no plans to add more storys;
4. (If applicable, i.e. it is not a bucket, as defined in §7.1.4) 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.

8.4 Handling Bugs & Emergent Work

8.4.1 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 §8.5, 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.

8.4.2 Issue Types

8.5 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 §4.1. 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 Data Management (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 §8.4.

8.6 Coordination Standup

9 Standard Reporting Cycle

Cathy ?

10 Personnel

10.1 Staffing Changes

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

- The Legacy Survey of Space and Time (formerly Large Synoptic Survey Telescope) (LSST) New Employee Onboarding material, and

and direct new recruits to them when they join your team[3].

The responsible hirere must also complete an onboarding form for the new recruit. When members of staff team leave the project, the T/CAM should fill in an offboarding form.

11 Open issues

- Kanban for LOE operations work
- Need section on more procedural driven work on mountain and DF.

A References

[LDO-31], Blum, R., et al., 2020, *LSST Operations Proposal* , LDO-31, URL `https://ls.st/LDO-31`

B Glossary

algorithm A computational implementation of a calculation or some method of processing.
B Byte (8 bit).
CA Control (or Cost) Account.
CAM CAMera.

[3] As per §5.2.1, remember that newcomers should be allocated SPs for working through this material.

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.

Data Management The LSST Subsystem responsible for the Data Management System (DMS), which will capture, store, catalog, and serve the LSST dataset to the scientific community and public. The DM team is responsible for the DMS architecture, applications, middleware, infrastructure, algorithms, and Observatory Network Design. DM is a distributed team working at LSST and partner institutions, with the DM Subsystem Manager located at LSST headquarters in Tucson.

Director The person responsible for the overall conduct of the project; the LSST director is charged with ensuring that both the scientific goals and management constraints on the project are met. S/he is the principal public spokesperson for the project in all matters and represents the project to the scientific community, AURA, the member institutions of LSSTC, and the funding agencies.

DM Data Management.

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.

JIRA issue tracking product (not an acronym but a truncation of Gojira the Japanese name for Godzilla).

LOE Level of Effort.

LSST Legacy Survey of Space and Time (formerly Large Synoptic Survey Telescope).

OBS Organisation Breakdown Structure.

Primavera The trade name for the project management software suite used by LSST to maintain its program plan and schedule.

Project Manager The person responsible for exercising leadership and oversight over the entire LSST project; he or she controls schedule, budget, and all contingency funds.

Review Programmatic and/or technical audits of a given component of the project, where a preferably independent committee advises further project decisions, based on the current status and their evaluation of it. The reviews assess technical performance and maturity, as well as the compliance of the design and end product with the stated requirements and interfaces.

Science Pipelines The library of software components and the algorithms and processing pipelines assembled from them that are being developed by DM to generate science-ready data products from LSST images. The Pipelines may be executed at scale as part

of LSST Prompt or Data Release processing, or pieces of them may be used in a standalone mode or executed through the LSST Science Platform. The Science Pipelines are one component of the LSST Software Stack.

seeing An astronomical term for characterizing the stability of the atmosphere, as measured by the width of the point-spread function on images. The PSF width is also affected by a number of other factors, including the airmass, passband, and the telescope and camera optics.

shape In reference to a Source or Object, the shape is a functional characterization of its spatial intensity distribution, and the integral of the shape is the flux. Shape characterizations are a data product in the DIASource, DIAObject, Source, and Object catalogs.

SP Survey Performance.

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.

T/CAM Technical/Control (or Cost) Account Manager.

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.

Work Breakdown Structure a tool that defines and organizes the LSST project's total work scope through the enumeration and grouping of the project's discrete work elements.