Rubin Observatory

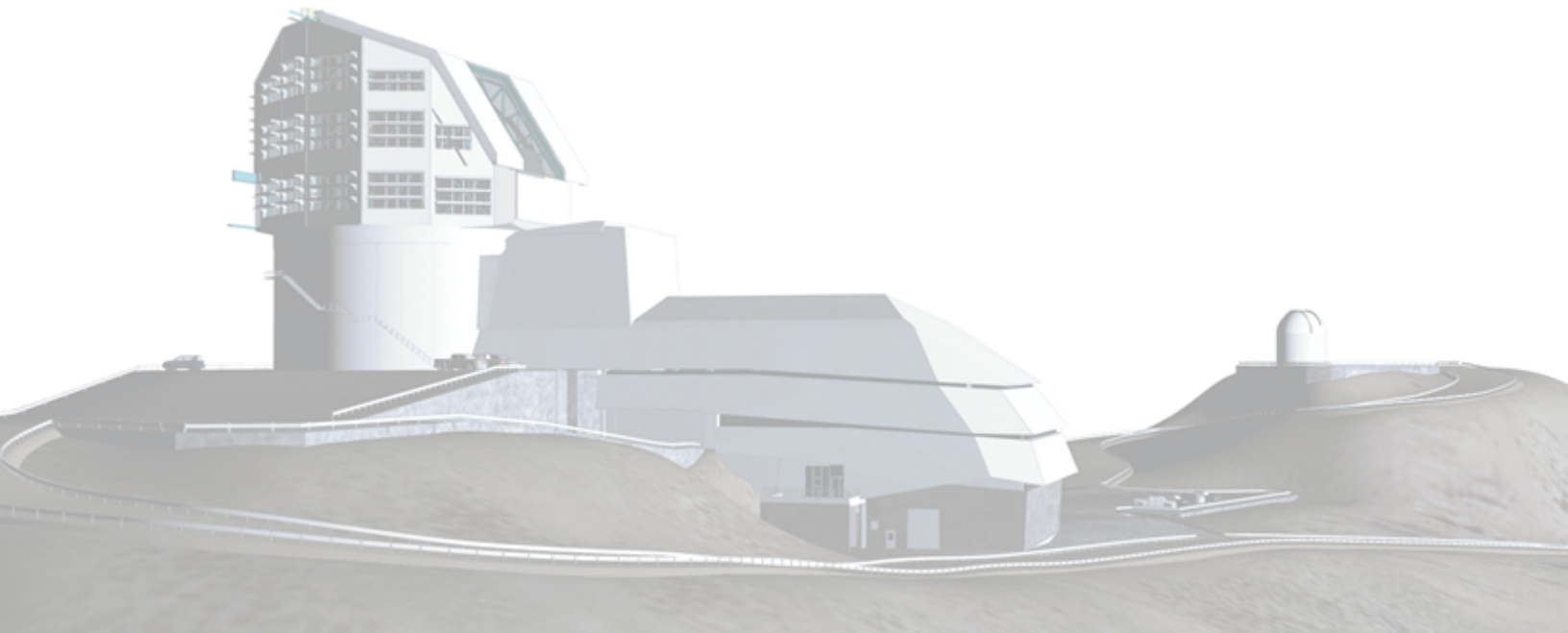
Vera C. Rubin Observatory
Data Management

Management principles and use of Jira for Rubin Operations

William O'Mullane, Amanda Bauer, Robert Blum, Phil Marshall

RTN-005

Latest Revision: 2020-11-04



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 Observatory approach to annual planning. 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, some within an Agile framework (particularly for data release planning). 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. To the extent possible, we focus on the people doing the work and how they work together.

This framework allows the multidisciplinary Rubin team to operate the facility and generate nightly data products while continuously improving efficiency of workflows ("Kanban" style[1]), as well as iteratively responding to user feedback on a longer timescale to maximize the scientific benefit of annual data releases ("scrum" style[2]).

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

[1] Kanban is a visual system for managing work as it moves through a process. Kanban is a concept related to lean and just-in-time (JIT) production, where it is used as a scheduling system that tells you what to produce, when to produce it, and how much to produce.

[2] https:en.wikipedia.orgwikiScrum_(software_development)

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 Rubin Operations Leadership and Annual Reporting

The Rubin Observatory Acting 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 Program Coordinator is Cathy Petry. She monitors the budgets and maintains details within WeBud, the NOIRLab budgeting system. She assists in developing the annual Program Operating Plan (POP), tracking milestones and reporting. On the SLAC side, Christine Soldahl is the Business Manager who handles similar tasks.

The POP is a defined report and process for NOIRLab. Rubin Operations considers the POP to be a Rubin activity, which informs both NOIRLab and SLAC leadership of the annual Rubin activity including milestones and budget. For NOIRLab, the Rubin POP is integrated and delivered to NSF for the next fiscal year at the end of the current fiscal year. For SLAC, the POP informs SLAC's annual planning, which culminates in a Field Work Proposal (FWP) for all SLAC High Energy Physics activity including Rubin.

The FWP is delivered in June of the current fiscal year, and covers the federal budget request for the next two fiscal years. The FWP is previewed to SLAC management and DOE in February in advance of the final delivery in June. Because the POP for NSF and FWP for DOE are out of phase, Rubin does high level planning in early Q2 of the financial year. Detailed activity planning, including lower level milestones, continues though the year in advance of the next year. This detailed planning is the subject of this document.

3 Formal Organizational Structure

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

3.1 Work Breakdown Structure

The Work Breakdown Structure (WBS) provides a hierarchical description of the activity based organization of Rubin. In operations, the WBS is not as formal as in a construction project, but it provides a useful structure to organize Rubin Operations and plan annual work around. Rubin is the level 1 of the WBS, the departments are level 2, and teams within the departments are level 3. Individual roles on operations are level 4.

Thus:

WBS	Description
1	Rubin Director's Office
2	Rubin Observatory Operations
3	Rubin Data Production
4	Rubin System Performance
5	Rubin Education and Public Outreach

Detailed work is planned in advance of each financial year at the team level. Team leads will work with department associate directors to develop plans for activities that address specific milestones, projects, and level of effort activity. Highest level milestones are reported regularly throughout the fiscal year to SLAC, AURA, NSF and DOE.

3.2 Level of Effort Work

There is work throughout Rubin Operations that will be recorded as a Level of Effort (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. LOE work is assumed to earn value (infor-

mally) simply through the passage of time.

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

As an example, a 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. Technical staff in Chile at the summit facility may spend a much more significant fraction of time as LOE. However, as above, we generally aspire to minimize the fraction of LOE for development activity.

4 Estimating Effort

4.1 Basic Assumptions

Rubin Operations 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 1,800 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. Rubin expects to pay the full rate for an individual with research time who contributes 1,440 hours to operations. This is handled through indirect rates at both NOIRLab and direct charges to research accounts at SLAC. Science time is included in the subcontracts of our partners at affiliated institutions through indirect charges similar to the

	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.

case for NOIRLab.

In Data Production, the 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[3], ad-hoc discussions and other interruptions. This work is counted as LOE. 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. For other staff, the LOE will be higher but include much more day to day activity than for the developer case.

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.

4.2 Special Cases

[3]"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 System PerFormances (SPs).

4.2.1 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 operations can be ticketed as regular work. For example, working through tutorials, reading documentation, and so on are all activities which can earn SPs.

4.2.2 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.

5 Long Term Planning

The authoritative, high-level summary of the long-term planning system may be found in any POP process document.

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 of a major aspect of Rubin Operations or the availability of specific functionality. Milestones are officially defined in a JIRA milestone issue.
2. A series of *epics* describe major pieces of work. Epics are associated with concrete, albeit high-level, deliverables or outcomes that culminate in the achievement of the above milestones, and have specific resource loads (staff assignments story point values) end dates. All epics are linked to the milestone they are created to help deliver, although some epics might exist without linking to a milestone (level of effort epics, for example).
3. A visualization of progress on work done towards achieving milestones is captured in

Smartsheet, which directly tracks progress by rolling up issues that are completed inside of JIRA epics that work together to deliver a given milestone.

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

Level 1 These are at the full observatory level and owned by the Directors Office. Examples are the completion of a Data Preview, the beginning of nightly observations for the full survey, or the delivery of an annual Data Release. Level 1 milestones are achieved by the culmination of effort defined by a set of Level 2 and Level 3 milestones. Level 1 milestones are reported to the agencies.

Level 2 These reflect effort within a Department and are owned by an Associate Director, or are cross-Department commitments. As such, they must be defined in consultation with the Director's Office. Level 2 milestones are achieved by the culmination of effort defined by a set of Level 3 milestones. Some Level 2 milestones may be reported to the agencies as defined by the annual POP.

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 (usually reserved for Level 1 milestones): it is important that these be delivered on time and to specification. Level 1 and 2 milestones are under change control once they are defined and described in a JIRA Milestone issue. Level 3 milestones are defined for use within Departments and not required to go under project change control, but properly adhering to the plan is important: 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 work to achieve milestones i.e. they may be blocking issues on the milestones. When a detailed description of work for a given epic is known, it is described in JIRA. It should then be assigned to the appropriate cycles.

Progress is tracked toward achieving milestones in Smartsheet by monitoring completed story points in JIRA epics and rolling up the total progress. To ensure success, JIRA epics must be completely detailed out prior to a full 6-month cycle and total effort should be estimated out for an entire fiscal year of effort, as detailed below.

5.1 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.

5.2 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.

6 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 are confirmed by the Director's Office, listed in Smartsheet, and detailed in the Milestone issue. Any team member can find the milestones in Jira.

6.1 Defining The Plan

6.1.1 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 (§5), 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 program 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 (§4) with the sum of the SPs in the epics you have scheduled (§6.1.2), 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.

6.1.2 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 this minimum set of fields filled in:

- A concrete, well defined deliverable *or* be clearly described as a "bucket" (§6.1.4);
- The `Component` field set to the appropriate Department;
- The `Due Date` field set to the appropriate date, which does not exceed the due date of the Milestone it is labeled to achieve.
- The `Story Points` field set to a (non-zero!) estimate of the effort required to complete the epic in terms of SPs (see §4).
- The `RO Milestone ID` field set to the appropriate L3 or L2 milestone that the epic will contribute to achieving (or achieve in its totality). The exact `RO Milestone ID`text can be found in the Milestone issue or Smartsheet.

The fields above are required to have values entered because they define the connection to Smartsheet where effort-tracking for the full project is done. Other fields in the epic can also be filled in as needed.

Be aware that: (Amanda: I don't think all of the following bullets are true or necessary in the Operations non-EVMS era. Should clarify with Cathy and Wil.)

- An epic may only be assigned to a single cycle. It is not possible to define an epic that crosses the cycle boundary (see §6.2 for the procedure when an epic is not complete by the end of the cycle).
- 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.

6.1.3 Scheduling Research Work

As discussed in §5.1, 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.

6.1.4 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.

6.2 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 (§6.1.4), which are, by their nature, timeboxed within the cycle).

7 Execution

Having defined the plan for a cycle following §6, we (RDP and RPF) 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.

7.1 Defining Stories

Epics have already been defined as part of the cycle plan (see §6.1.2). 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 (§7.2) before the sprint starts. When doing so, you may wish to leave some spare time to handle emergent work (discussed in §7.4).

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.

7.2 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 (§7.1). 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 §4).

   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 effort toward milestones.

   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).

7.3 Closing Epics

7.3.1 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 §6.1.4) 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.

7.4 Handling Bugs & Emergent Work

7.4.1 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 §7.5, 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.

7.4.2 Issue Types

7.5 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 §3.1. 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 §7.4.

7.6 Coordination Standup

8 Tracking Progress and Standard Reporting Cycle

Cathy ?

8.1 Tracking Progress toward Milestones

Progress on completing epics is visualized in Smartsheet. Smartsheet lists all Level 1 through Level 3 milestones in a gantt-chart style view. Each Level 1 milestone is achieved by completing a series of Level 2 and/or Level 3 milestones. Smartsheet tracks Story Points marked as complete in individual JIRA epics in real time. Progress on individual milestones is shown as the weighted total of Story Points within each epic contributing to the successful completion of the milestone.

8.2 Reporting Cycle

High level milestone progress will be reported to SLAC and NOIRLab regularly. NOIRLab reports will flow quarterly (or monthly) to the NSF. Rubin will show progress on all L1 milestones and any L2 milestones called out in the POP.

9 Personnel

9.1 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[4].

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.

10 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.

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.

[4] As per §4.2.1, remember that newcomers should be allocated SPs for working through this material.

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.

Earned Value Management System A set of tools, techniques and procedures which are used to implement a EVM approach to project 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).

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.

SP System 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 SP; beyond that, the work is insufficiently fine-grained to schedule as a story. While fractional SP are fine, all stories involve work, so the SP 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.