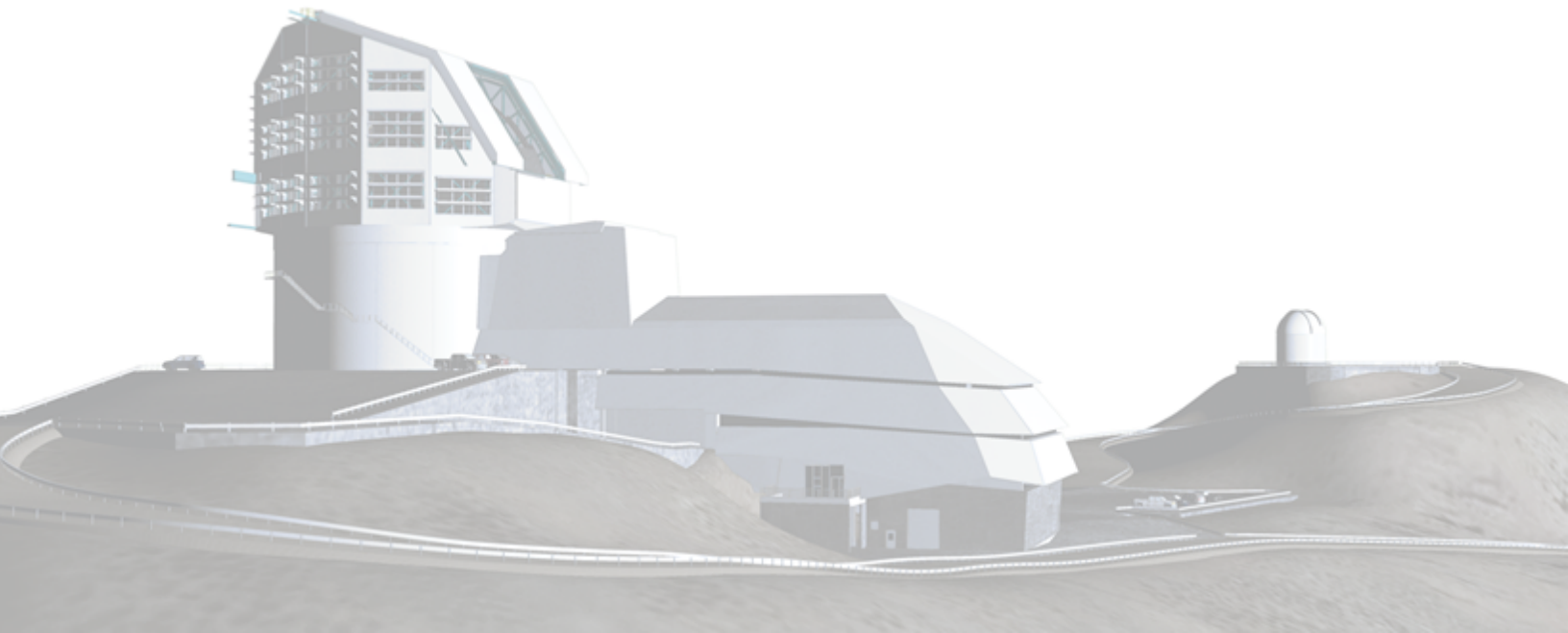

Vera C. Rubin Observatory
Data Management

Work Management Systems for Rubin Operations

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

RTN-005

Latest Revision:

Abstract

This document describes guidelines for the management of effort in Vera C. Rubin Observatory Operations. It describes the process for planning and executing 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 activity. This document describes how effort towards an annually-planned schedule is tracked.

Change Record

Version	Date	Description	Owner name
1	2021-11-05	Unreleased.	William O'Mullane

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

Contents

Work Management Systems for Rubin Operations

1 Introduction

This document provides a guide to the Vera C. Rubin Observatory approach to work management and annual planning. See the operations proposal RDO-018 for a description of the full scope and high level goals of the program. There is no formal Earned Value Management System (EVMS) required from the funding Agencies (the National Science Foundation (NSF) and the Department of Energy (DOE) Office of Science) so activities are planned in detail on an annual basis and effort towards that schedule of activities is tracked through an agile process.

The annual planning process starts by setting high level milestones for the year, which are centered around releasing data to the public and major maintenance to the telescope system once we enter the phase of full survey operations. The Leadership Team builds a series of milestone activities that are discrete pieces of work within Departments and Teams to collectively deliver the high level milestones. Teams record their day to day work in JIRA and overall progress is monitored automatically through Smartsheet and reported to the Agencies through our managing organizations.

This framework allows the multidisciplinary Rubin teams to operate the facility and generate nightly data products while continuously improving efficiency of workflows, as well as iteratively responding to user community feedback on a longer timescale to maximize the scientific benefit of annual data releases. Examples include optimizing the observing strategy as the survey progresses, improving algorithms in response to the user community feedback, 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 annual plans, effectively track work progress to ensure delivery of milestones, maintain visibility in our workflows, remain responsive to change, and offer staff the ability to innovate and collaborate.

2 Organizational Structure

2 Organizational Structure

2.1 Rubin Operations Leadership

Rubin Observatory is a Program of NSF's NOIRLab. The Rubin Observatory Director is Robert Blum, the 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 Operations.

The Head of Operations is Ranpal Gill and the Program Coordinator is Cathy Petry. They monitor the budget and maintain details within the NOIRLab accounting system. They assist in developing the annual Program Operating Plan (POP), tracking milestones and reporting on progress.

On the SLAC side, Christine Soldahl is the Business Manager who handles similar tasks.

Rubin Operations has four operational Departments in addition to the 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 led by an Associate Director (AD).

2.2 Annual Reporting

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 (January and February). Detailed activity planning, including defining smaller chunks of work as lower level milestones,

continues though the year in advance of the next year. This detailed activity planning is the subject of this document.

2.3 Work Breakdown Structure

The Work Breakdown Structure (WBS) is a hierarchical description of Rubin Operations from an activity-based perspective. It provides a useful structure to organize Rubin Operations and plan annual work around. Rubin is level 1 of the WBS, the departments are level 2, and teams within the departments are level 3 and in the case of Program Operations, level 4. Individual roles in operations are defined at the lowest level.

This table shows the level 2 and level 3 elements of the Rubin Operations work breakdown structure.

L2 WBS	L3 WBS	Description
1		Rubin Director's Office
	1.1	Director's Office
	1.2	Safety
	1.3	Program Operations[1]
	1.4	In-Kind Program Coordination
	1.8	Legacy Survey of Space and Time
	1.10	Sustainability
	1.11	Site Protection
2		Rubin Observatory Operations
	2.1	Observatory Operations Management
	2.2	Observatory Science Operations
	2.3	Observatory Software
	2.4	Summit Operations
	2.5	Nighttime Operations
	2.6	Engineering
3		Rubin Data Production
	3.1	Data Production Management
	3.2	Infrastructure and Support
	3.3	Data and Processing Architecture
	3.5	Algorithms and Pipelines
	3.6	Service Quality and Reliability Engineering

L2 WBS	L3 WBS	Description
	3.7	DevOps Support
	3.8	Data Security
4		Rubin System Performance
	4.1	System Performance Management
	4.2	Verification and Validation
	4.3	Community Engagement
	4.4	Survey Scheduling
	4.5	Systems Engineering
5		Rubin Education and Public Outreach
	5.1	EPO Management
	5.2	EPO Technical
	5.3	Education
	5.4	Outreach

Detailed work is planned in advance of each fiscal 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. Progress towards the highest level milestones is reported regularly throughout the fiscal year to SLAC, AURA, NSF and DOE.

2.4 Activity Types

There are two types of activities (or epics) that are planned: activities that result in a deliverable, and level of effort (or support) activities. Progress can be tracked on activities with a deliverable by computing the fraction of the effort that is complete (percent complete) based on completed stories linked to the epic. Progress on Level of Effort (LOE) work is assumed to progress proportionally with the passage of time.

LOE activities include attending meetings, reporting on milestones, or taking part in other activities which do not directly map to a deliverable or product. This may be particularly the case for technical managers or others in leadership roles. In general, we strive to minimize the fraction of effort which is devoted to LOE activities and favor those which are more directly

[1] Program Operations is made up of several groups at level 4 that are not presented here but are available for activity planning and budgeting.

accountable. However, in certain cases such as operations and maintenance of telescope and facility systems, pipelines or other systems, LOE is perfectly acceptable. Technical staff in Chile at the summit facility may spend a much more significant fraction of time as LOE. 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 planned and tracked against well-defined deliverables.

3 Estimating Effort

3.1 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” or “engineer” roles can allocate up to 2% of their time to training, 2% to administrative activities and 1% to outreach and DEI activities.

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 a NOIRLab’s Research and Science Services (RSS) project code and is prorated for staff who are fractionally allocated to Rubin Operations. Training, administrative, and outreach and DEI time is charged to either RSS or ES depending on where staff will move into NOIRLab (RSS or ES). Functional Managers will ensure the proper project codes are available on your timecard.

Rubin expects to pay the full rate for any scientist or engineer who contributes full-time or fractionally to operations. This is handled through indirect rates at NOIRLab and direct charges to research accounts at SLAC. Science time is included in the subcontracts of our partners at

	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.

affiliated institutions through indirect charges similar to the 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[1], 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.

3.2 Special Cases

[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 System PerFormances (SPs).

3.2.1 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 to "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.

3.2.2 Team Leads 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.

4 Long Term Planning

4.1 Components of Annual 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, availability of specific functionality, or maintenance event for the telescope system. Milestones are planned in Smartsheet and then 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) and

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 or emergent work 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 are 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. Note the change control process is under development as a Pre-Operations activity.

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 on linked issues 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. All milestones should appear in JIRA with a milestone issue type as the source of truth.

4.2 Planning Research Work

In order for Rubin Observatory 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.

4.3 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.

4.4 Defining the Schedule with Milestones

Rubin Milestones are defined as JIRA issues of type "milestone". As indicated above, the Director (or their designate) defines the L1 milestones, the Associate Directors (ADs) define their departments' L2 milestones, and the Team Leaders and ADs define the L3 milestones for their teams.

L1, L2 and most L3 milestones are defined as part of the annual planning cycle, and prior to the year in which the work associated with them is due to be carried out. ADs and Team Leaders communicate their milestones to the Program Coordinator, who enters them into

Smartsheet and then creates a JIRA issue of type “milestone” for each one.

During the year, it is sometimes necessary to create new milestones (primarily at Level 3) that were omitted during the earlier planning phase. In this case, the team leader or AD may create the JIRA milestone directly, and alert the Program Coordinator to it for inclusion in the Smartsheet.

The following fields must be filled out when defining a new milestone:

- The `Milestone Level`, “1,” “2,” or “3.”
- The `RO Milestone ID`. This is a string like “L2-DI-0003” which indicates the level of the milestone, the department (for level 2 milestones) or team (for level 3 milestones) that owns it, and an ID number that serves to make the ID unique but which otherwise carries no meaning. The list of two-character department and team string identifiers can be found in the Appendix (“DI” in the example stands for “Directorate”). The ID number may get edited by the Director’s Office to ensure that it is unique: the Milestone IDs are only ever used in linking together the milestones for visualization in Smartsheet. Good practice here is to look at the milestone list in Smartsheet and choose an ID number that leads to a milestone ID that has not already been taken, and which roughly fits the milestone into the time-ordered list.
- The `Summary` field is equivalent to the activity that will take place in order to produce the deliverable and meet the milestone. It should contain a sentence outlining the activity to be performed in order for the milestone to be reached and the deliverable to be produced. In the POP document tables this is the ”Activity”. Examples include *“Deliver Data Preview 0.1 (DP0.1)”* (an L1 milestone) and *“DP0.1 Data Release: science-ready catalogs released from the IDF”* (an L2 milestone that belongs to it).
- The `Description` text should contain more information detailing the scope of activity needed to complete the milestone. Example: *“Upgrade the WBS activity, labor and non-labor plans from V4 to V5 in order to capture a US DF at SLAC, a UK DF, and any other modifications needed, and estimate the corresponding budget.”* Note that only L1 (and sometimes L2) milestones are actually listed in the NOIRLab Program Operations Plan (POP), but Rubin adopts the same structure for all its milestones.
- The `POP Milestone` is concise description of the when the milestone is reached. Example: *“V5 WBS workbook and Preliminary Cost Calculator implementation complete.”*

- The `POP Deliverable` is a very terse list of the deliverables needed to reach the milestone. Example: *"V5 WBS workbook and Preliminary Cost Calculator."*
- The `Due Date` is the latest date in the future by which the milestone needs to be reached. This date should be before or the same as the milestone's parent milestone's "Rubin Forecast Due Date" as shown in the Smartsheet.
- The `Start Date` is the date when the work for the milestone should begin. This is the date that the Smartsheet will use in a visual comparison between the fraction of work completed and the fraction of time elapsed, to help track progress on the epics.

The Program Coordinators will ensure that the milestones that have been defined are correctly linked together in the Smartsheet, so that their epics appear nested beneath them.

5 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.

5.1 Defining The Plan

5.1.1 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 (§4), 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 (§3) with the sum of the SPs in the epics you have scheduled (§5.1.2), 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.

5.1.2 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" or "emergent work" (§5.1.4);
- The `Component` field set to the appropriate Department;
- The `Story Points` field set to a (non-zero) estimate of the effort required to complete the epic in terms of SPs (see §3).
- 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` can be found in the milestone issue or Smartsheet.
- 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 `label` field is set to identify the fiscal year during which the work will be done. Examples are FY22 or FY23.

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:

- An epic may only be assigned to a single cycle. It is not possible to define an epic that crosses the cycle boundary (see §5.2 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 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.

5.1.3 Scheduling Research Work

As discussed in §4.2, 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.

5.1.4 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.

5.2 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 (§5.1.4), which are, by their nature, timeboxed within the cycle).

Be aware that if a planned epic is not closed it may impact the completion of the milestone it contributes to. Epics related to milesteones must be completed in order for the milestone to also be completed.

6 Execution

Having defined the plan for a cycle following §5, 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.

6.1 Detailing Work

6.1.1 Issue Types

There are two JIRA issue types that are used for planning work on epics: Story and Bug

6.1.2 Defining Stories

Epics have already been defined as part of the cycle plan (see §5.1.2). 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 (§6.2) before the sprint starts. When doing so, you may wish to leave some spare time to handle emergent work (discussed in §**??**). Every epic should contain *at least* one story.

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. After execution is complete, comparing the total number of SP on planned stories in an epic to the number of SP on the epic itself affords the opportunity to refine time estimates going forward.

6.1.3 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 §6.4, 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.

6.1.4 Emergent Work

On occasion, work arises that is not anticipated and therefore not planned. Epics for this type of work will have been set up every cycle so stories should be linked there.

6.2 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 (§6.1.2). 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 §3).

   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 (see §6.5), 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 progress 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 Team Lead, 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 Team Lead 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).

6.3 Closing Epics

An epic is considered complete and 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 §5.1.4) its concrete deliverable has been achieved.

"Bucket" epics should be closed at the end of the time box (i.e. end of half fiscal year and end of fiscal year). Note that it is not permitted to close an epic without defining at least one story within it. Empty epics can never be completed.

6.4 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 §2.3. Where there is uncertainty about which team should be responsible for a particular ticket, the "Data Production Management" team may be used to indicate that the Associate Director (AD) of Data Production 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 §**??**.

6.5 Coordination Standup

7 Tracking Progress and Standard Reporting Cycle

7.1 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.

7.2 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.

8 Personnel

8.1 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[2].

The responsible hirere must also complete an onboarding form for the new recruit. When members of staff team leave the project, the Technical/Control (or Cost) Account Manager (T/CAM) should fill in an offboarding form.

9 Open issues

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

A References

[2] As per §3.2.1, remember that newcomers should be allocated SPs for working through this material.

B Glossary

AD Associate Director.

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.

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.

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).

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.