LARGE SYNOPTI C SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)

Telescope and Site Software Organization and Management

William O'Mullane and Andy Clements

LTS-928

Latest Revision: 2019-06-19

Change Record

Version	Date	Description	Owner name
1.0	2019-06-19	Initial version (Document-928).	A. Clements

Document source location: TBD

Contents

Telescope and Site Software Organization and Management

1 Introduction

1.1 Purpose

This document defines the mission, goals and objectives, organization and responsibilities of the LSST Telescope and Site Software subsystem ("TSSW"). The document is currently scoped to define these elements for the LSST Design, Construction, and Commissioning phases. It does not address any ongoing mission for TSSW during LSST Operations.

1.2 Mission Statement

Create operable, maintainable, high-quality software products for controlling, monitoring and maintenance of the telescope and its facility, all on time and within reasonable cost.

1.3 Goals and Objectives

LSST Telescope and Site Software will:

- Define the subsystem products, data logging mechanisms, data interface and curation requirements for LSST (with approval by others).
- Assess current and operations-era technologies for use in providing engineered solutions to the requirements.
- Select, implement, construct, test, document, and deploy the software infrastructure, middleware, applications, and external interfaces.
- Document the operational procedures associated with using and maintaining software capabilities.
- Evaluate, select, recruit, hire/contract and direct permanent staff, contract, and in-kind resources in LSST and from partner organizations participating in LSST software initiatives.

The TSSW goals in developing LSST software solutions are:

- Acquire and/or develop solutions: To achieve its mission, LSST TSSW prefers to acquire and configure existing, off-the-shelf, solutions. Where no satisfactory off-the-shelf solutions are available, TSSW develops the software systems necessary to:
  - Enable the generation and deployment of LSST telescope control products at the LSST Summit and Base Facility
  - Enable the automatic control and monitoring of the LSST Telescope from multiple places.
  - Enable the communication of each software component to any other software component through a publish/subscribe interface
  - Have the ability to store and query all telemetry, events & commands for a 30 day period at the summit and for the entire survey at the base.

- Maintain coherent architecture: TSSW software architecture is actively managed at the subsystem level. A well-engineered and cleanly designed codebase is less buggy, more maintainable, and makes developers who work on it more productive. Where there is no significant impact on capabilities, budget, or schedule, LSST TSSW prefers to acquire and/or develop reusable, open source, solutions.
- Support reproducibility and insight into algorithms: Other than when prohibited by licensing, security, or other similar considerations, TSSW makes all newly developed source code, and in particular that pertaining to scientific algorithms, public. Our primary goals in publicizing the code are to simplify reproducibility of LSST telescope control products and to provide insight into algorithms used. Achieving these goals requires that the software be properly documented.
- Opportunities beyond LSST: the LSST TSSW codebase may be of interest and (re)used beyond the LSST project (e.g., by other telescopes). While enabling or supporting such applications goes beyond LSST's construction requirements, cost and schedule-neutral technical and programmatic options that do not preclude them and allow for future generalization should be strongly preferred.
- Background decision material on choices made in TSSW will be documented in technical notes which will be lodged in DocuShare (see Section 4.10 & 4.11) with "LTS" series handles.

2 Telescope and Site Conceptual Architecture

The TSSW Subsystem Architecture is detailed in LSE-150. A few of the higher level diagrams are reproduced here to orient the reader within TSSW.

During Operations, the control components of the TSSW Subsystem will be installed and run at the summit computer room at Cerro Pachon. This telescope is designed to be an autonomous system - running the survey all night with little interaction from the operator. This complex hierarchy of computer systems is divided out into several namespaces:

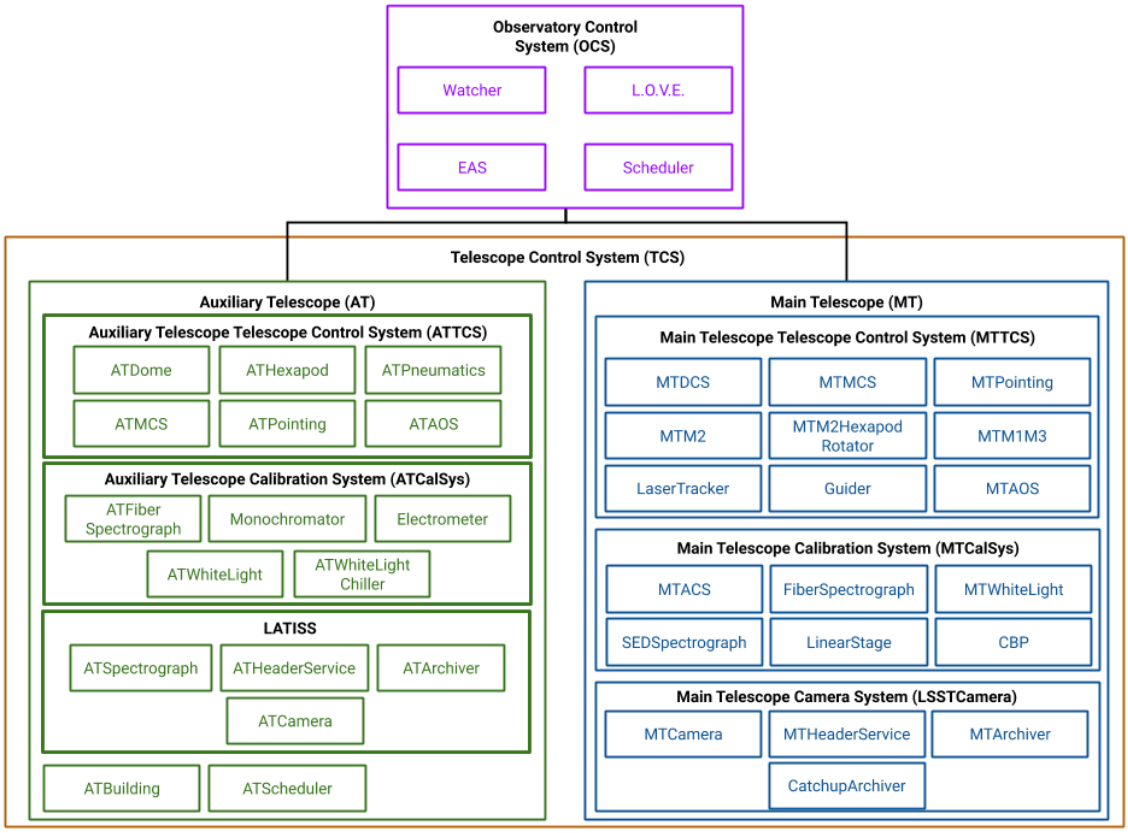

Figure 1

Figure 1 shows the various TSSW namespaces and which current components each will contain. The Observatory Control System (OCS) allows for operation personnel to interact with the telescope and the telescope's environment. This contains the scheduler for deciding the survey for the night, the L.O.V.E. (LSST Operator's Visualization Environment) for monitoring of activities and interaction with the telescope for triaging problems, the Watcher for monitoring alerts and the EAS (Environmental Awareness System) for monitoring and controlling the telescope

environment. These components will be used in controlling and monitoring the Telescope Control System (TCS). The TCS can be subdivided further into the Auxiliary Telescope and Main Telescope. There are also several groupings of components within each of these namespaces. Most components communicate through the middleware called Service Abstract Layer (SAL). Telemetry, Event and Command storage and transport between components is provided by SAL. SAL is an in-house wrapper for the Vortex OpenSplice Data Distribution Service (DDS), a third party communications protocol technology provided by ADLink. This third party technology is one implementation of many of the DDS (Data Distributed System) specifications. The storage component of SAL is the Engineering Facility Database (EFD). This complex piece of infrastructure is displayed in Figure 2.

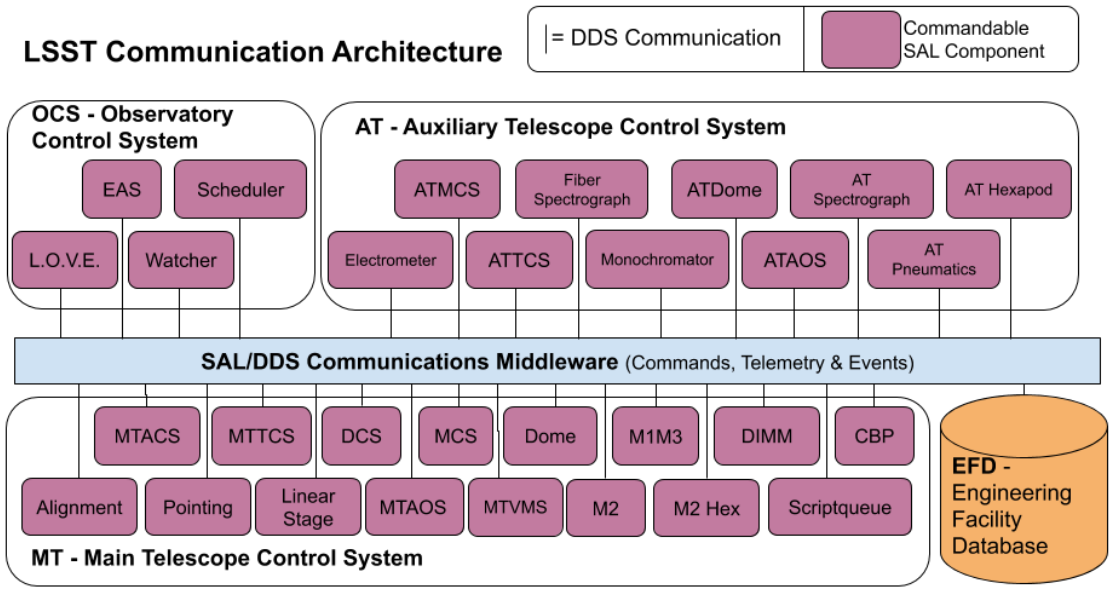

Figure 2

Figure 3 shows the common infrastructure and services layer which underlies the computing environments. This does list specific technologies for managing/monitoring, provisioning/deploying, or workload/workflow — however these are still being selected — but under consideration are industry-standard tools such as GitHub, Jenkins, Puppet, Kubernetes, Grafana, etc.

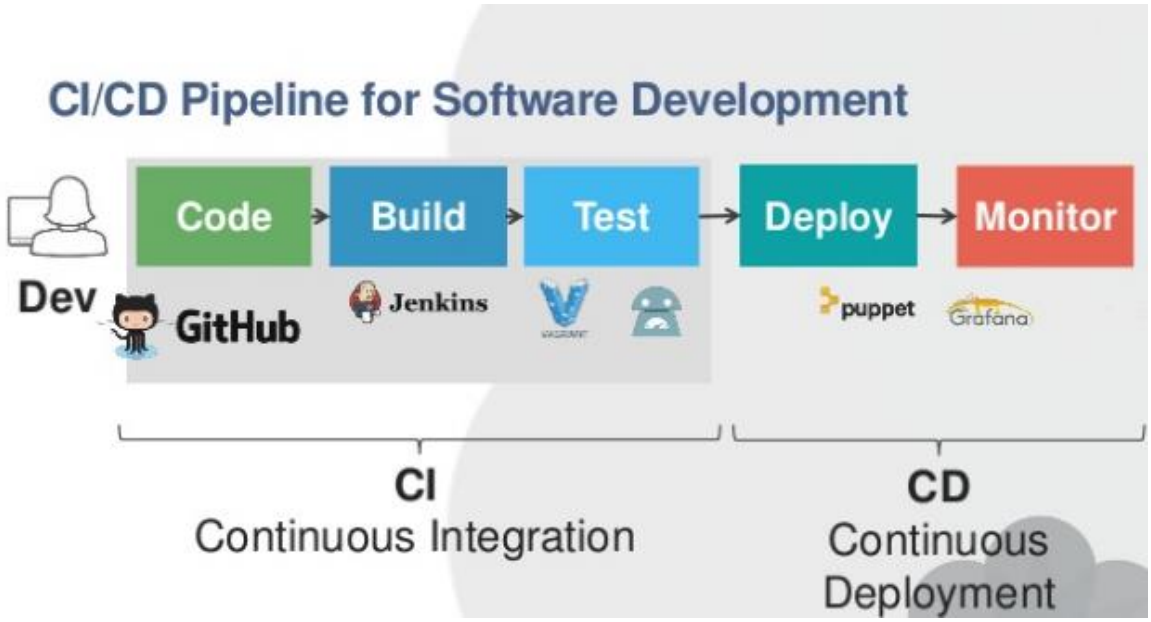

Figure 3

The TSSW can be commanded from multiple locations via the L.O.V.E. interface. These include:

- The Telescope itself at Cerro Pachon, Chile
- The Base Facility in La Serena, Chile
- The Simulation Cluster in Tucson Arizona
- Sending Monitor Information to any web-enabled device with authorized access.

2.1 External Interfaces & Auxiliary Data

The TSSW external interfaces are defined by the ICDs listed in Table 1.

LSE-66	Guider interface between the Camera and the Telescope
LSE-67	Wavefront Sensor Interface between the Camera and Telescope
LSE-70	LSST Observatory Control System Communication Interface
LSE-71	OCS-Camera Software Communication Interface
LSE-72	OCS - Data Management Software Communication Interface
LSE-73	OCS-Telescope Software Communication Interface
LSE-75	Control System Interfaces between the Telescope and Data Management

LSE-132	OCS to Summit Facility Monitoring Interface
LSE-140	Auxiliary Instrumentation Interface between DM & Telescope
LSE-209	Software Component to OCS interface Control Document

Table 1

3 Telescope and Site Software Organizational Structure

This section defines the organizational structure during the period in which the TSSW Subsystem software is developed and commissioned, up to the start of LSST Observatory operations.

The Software Deputy Project Manager leads the DM Subsystem and TSSW. The Telescope and Site Project Manager leads the Telescope and Site construction efforts at both the Telescope at Cerro Pachon and the Base Facility at La Serena. Both the Director of Software and Telescope and Site Project Manager have direct responsibility for coordination with the overall LSST Project Office, the LSST Change Control Board, the LSST Corporation, and LSST partner organizations on all budgetary, schedule, and resource matters.

TSSW Manager leads the TSSW group. The TSSW Manager is responsible for schedules, budgetary determination and resource allocation for the TSSW group. The TSSW Architect and Lead Developer help decide technical direction and overall architecture of the LSST Telescope and Site Control System. The Product Owners are scientists and system engineers, representing various disciplines, making sure the software produced is able to generate the science required from the telescope. The Product Owner state WHAT the software should produce, the architect and lead developer in conjunction with the developers will determine HOW it is produced.

FIGURE 4: TSSW organization.

3.1 Meetings

As part of a diverse organization, TSSW staff will participate in a considerable number of meetings. NSF and AURA have many rules on meeting attendance and LSST keeps policies updated accordingly in LPM-191 and Document-13760. This includes the travel summary report template [Document-13762] every traveler must fill after attending a meeting.

The TSSW Manager may, on occasion, require that travelers to a specific meeting provide a detailed debriefing note or presentation.

3.2 Working Groups

The regular decision making process within TSSW is based on individual empowerment and a mechanism to develop consensus. This Request for Comment, or RFC, process is described in Appendix C.

However, some issues in the development of a system that comprises many subsystems is that it requires more effort to reach resolution than can be reasonably addressed through an RFC. When required, the Director of Software will address these issues through the creation of a short-lived "working group". The working group will be given a specific narrow charge, it will be a small group (of perhaps around seven people), its activities will be bounded in time, and it will have a clear deliverable. Members of the group will be agreed by the TSSW Manager to provide the best technical input from the perspective of all stakeholders. Since most members of TSSW have their time scheduled in advance (following the procedure described in Section 4.3), it is important to consider the impact of working group activities on the overall TSSW schedule. In particular, the consent of the relevant T/CAM should be obtained before a member of their teams is added to a working group. Members of the working group should discuss in their local organizations and socialize recommendations ahead of adoption. The working group charge will be "RFC"ed in the usual manner to reach an agreed version and to broadly communicate the formation of the Workgroup. The RFCs for working groups are considered automatically flagged (i.e., not subject to self-adoption); typically, the Director for Software will adopt them by executive decision after consulting with the TSSW Manager. The adopted version of the charge will be issued as an LTS document.

3.3 External Studies

The Director of Software may initiate or request studies by external parties to investigate or report on technological or other choices facing the TSSW software.

3.4 Document Management

TSSW documents will follow the Systems Engineering Guidelines of LSST. PDF versions of released documents shall be deposited in DocuShare in accordance with the Project's Document Management Plan [LPM-51].

An "LTS-" prefix on a document handle indicates that the document is change-controlled at the subsystem level; i.e., it may be released or modified only with the agreement of the TSCCB[1] (Telescope and Site Change Control Board). Uncontrolled documents, such as technical notes

[1] https://confluence.lsstcorp.org/pages/viewpage.action?pageId=3113550

(prefix “Document-”), may be released whenever the author decides it is appropriate (or when a release is requested by the Director of Software).
The document tree for TS is shown in Figure 5. This is not exhaustive, but serves to give a high level overview of the main documents in TS and the relationships between them.

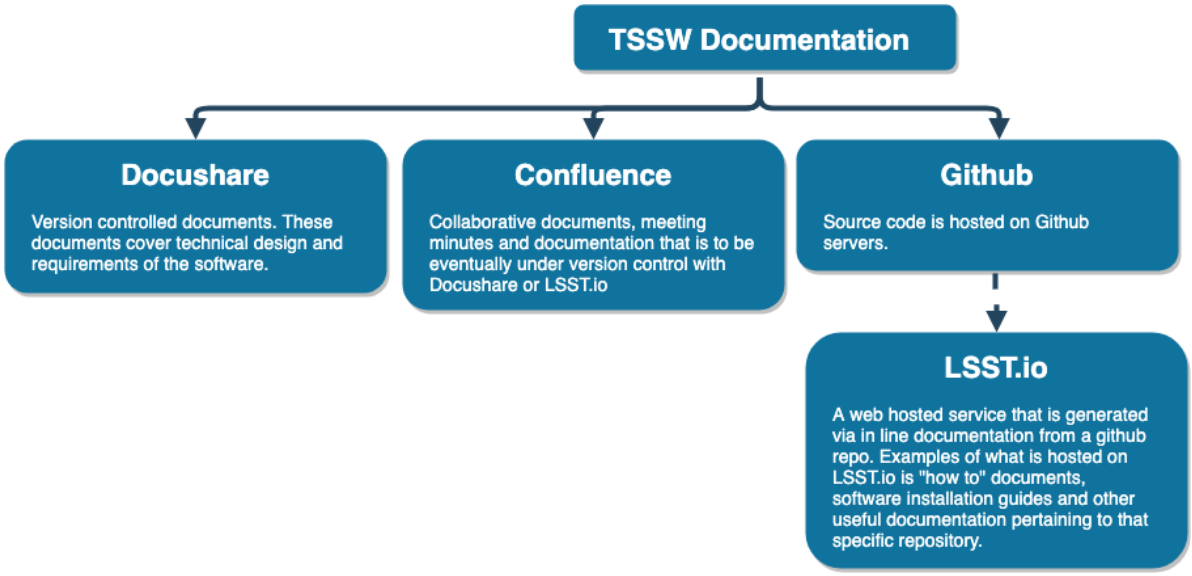

FIGURE 5: The document tree for the Telescope and Site subsystems.

3.4.1 Draft Documents

Draft TSSW documents will be kept in Confluence. Use of Google Docs or Confluence is tolerated but final delivered documents must conform to the standard LSST format, and hence either produced with LaTeX, using the lsst-texmf package[2], or Word, using the appropriate LSST template [Document-9224, Document-11920]. The precursor document should then be erased with a pointer to the baseline document, stored in Docushare.

[2] https://lsst-texmf.lsst.io/

3.4.2 End-User Documentation

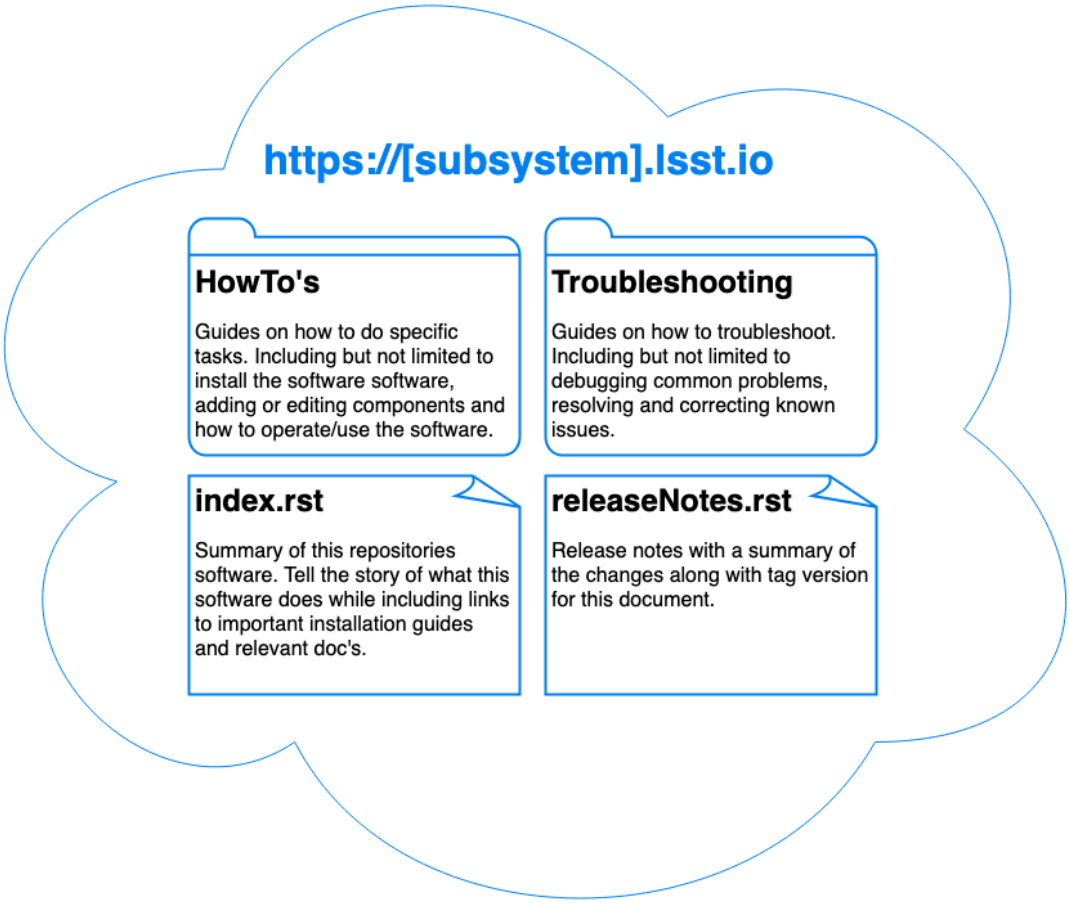

FIGURE 6: Outline of the web hierarchy for the TSSW end user documentation.

As shown in Figure 6, end-user documentation has been subdivided into four folders, HowTos, Troubleshooting, Repository indexes & Release Notes. Appropriate web-based, user-focused documentation is regarded as a major TSSW deliverable.

3.5 Configuration Control

Configuration control of documents is addressed in Section 3.4. In this section, we consider instead how configuration control is applied to operational systems and software development.

3.5.1 Software Version Control

TSSW follows a git based versioning system based on public git repositories. The approach is covered in the Developer Guide[3] and is consistent with the Project-level Systems Engineering

[3]https://tssw-developer.lsst.io/v/couger01-feature-initial_port/index.html

Management Plan LSE-17. The master branch of the repository will be used for production - the current version running the telescope. With the develop branch attempts are made to have stable code with the development work being done in the offshoot branches (named with the ID of the corresponding Jira Task describing the work). Once reviewed, an offshoot branch is unit tested and merged to develop. It should be functional and releasable for test. Releases are recorded by tagging the master branch; hot-fix branches are created if patches are required in the master branch.

As we approach commissioning and operations TSSW will have much stricter version control. At this point there will be a version of the software which may need urgent patching, a next candidate release version of the software, and the master. A patch to the operational version will require the same fix be made in the two other versions. The role of the TS Change Control Board (TSCCB) becomes very important at this point to ensure only essential fixes make it to the live system as patches and that required features are included in planned releases.

We cannot escape the fact that we will have multiple code branches to maintain in Commissioning which will lead to an increase in workload. Hence one should consider that perhaps more manpower may be needed in commissioning to cope with urgent software fixes while continuing development. The other consideration would be that features to be developed post commissioning will probably be delayed more than one may think, as maintenance will take priority.[4]

3.5.2 Hardware Configuration Control

On the hardware side we have multiple configurable items; we need to control which versions of software are on which machines. These days tools like Puppet make this reasonably painless. Still the configuration must be carefully controlled to ensure reproducible deployments providing correct and reproducible results. The exact set of released software and other tools on each system should be held in a configuration management database. Changes to the configuration should be endorsed by the TSCCB.

3.6 Risk Management

Risks will be dealt with within the LSST Project framework as defined in LPM-20. Risks in TSSW may be sent to the Director for Software at any time for consideration to be included in the formal risk register (appropriately costed and weighted). All risks are reviewed regularly by the Director for Software and the Systems Engineer (minimum every 3 months).

3.7 Quality Assurance

In accordance with the project QA plan [LPM-55] we will perform QA on the software products. This work will mainly be carried out by the Quality Assurance Lead (Section 6.7). Quality Assurance here means compliance with project guidelines for production, in our case for software production. A part of this is to have a verification/validation plan(s) which in and of itself is a major task (see Section 3.9).

3.8 Action Items

Actions in TSSW are tracked as Jira issues and periodically reviewed at TSSW meetings.

[4] WOM identifies this as the maintenance surge.

3.9 Verification and Validation

We intend to verify and validate as much of TSSW software as possible before commissioning and operations. This will be achieved through testing and operations rehearsals/data challenges. The verification and validation approach is detailed in LTS-760, which includes a detailed discussion of the test schedule summarized in Figure 7.

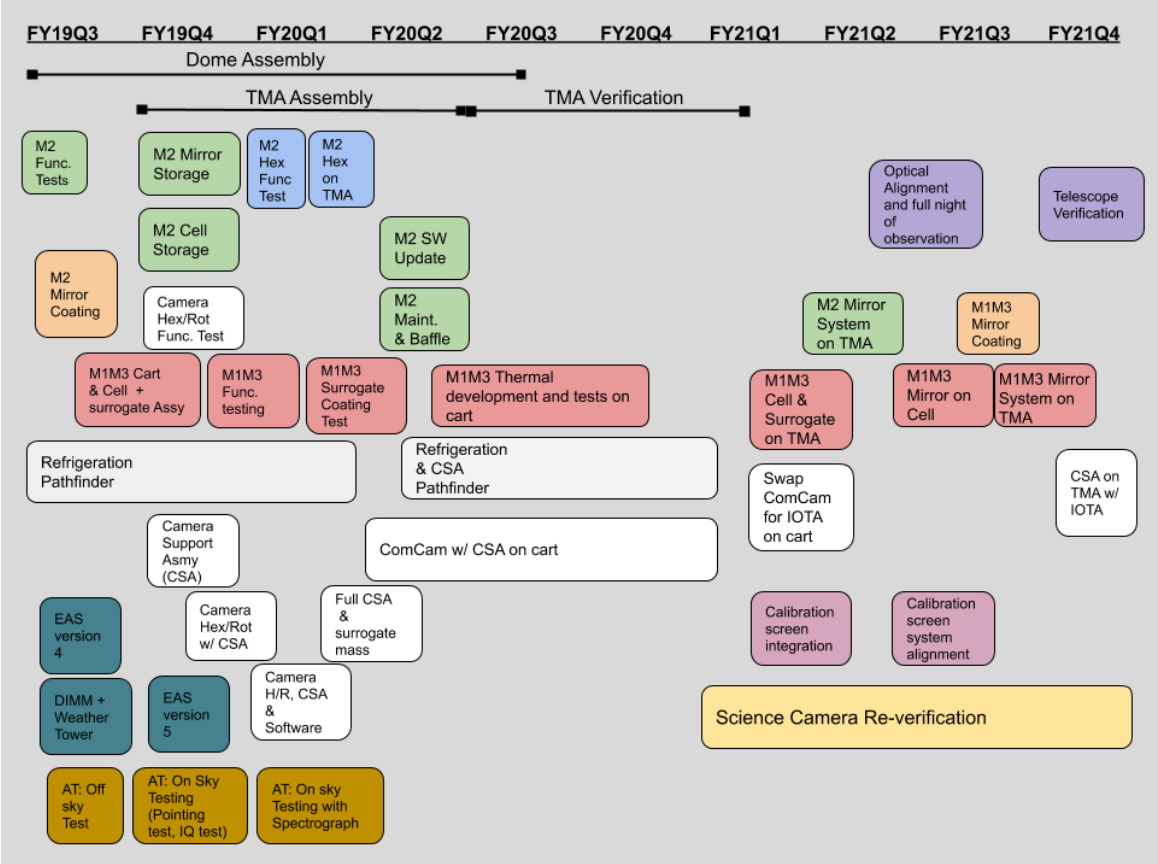

Figure 7 - Assembly, Integration & Verification (AIV) Process Schedule

4 Project controls

TSSW follows the LSST project controls system, as described in LPM-98. Considerations specific to TSSW are outlined in Section 4.3.

The TS Software Manager is responsible for the PMCS (Project Management and Control System) and, in particular, for ensuring that TSSW properly complies with our earned value management requirements. The Manager is the first point of contact for all questions about the PMCS.

4.1 Schedule

The entire LSST project schedule is held in Primavera. Tied to major project milestones we have a series of TSSW tests which need to be performed to show readiness for the different project phases. The project schedule is updated according to the demands of the project every 3 to 6 months. An example of this schedule can be seen in Appendix F.

4.2 Work breakdown structure

While the original Telescope and Site WBS is laid out in LPM-43 with definitions provided in LPM-44, the current WBS is detailed in Appendix B.
The WBS provides a hierarchical index of all hardware, software, services, and other deliverables which are required to complete the LSST Project. It consists of alphanumeric strings separated by periods. The first component is always "1", referring to the LSST Construction Project.

4.3 Planning Process

Milestones have been defined to describe the major goals of the TSSW subsystems for the next 3 to 6 months of the project. New milestones are added each 3-6 months depending on the work needing to be done. Each milestone has a description, a level and an associated JIRA epic. Four levels are defined:

Level 1	The most important milestones exposed at the NSF level.
Level 2	Cross-subsystem milestones (for example, TSSW milestones that affect the Camera Subsystem).
Level 3	Cross-team milestones within TSSW (for example, Middleware milestones that affect the TMA Team).
Level 4	Internal milestones within a team.

Table 2: TSSW top level Work Breakdown Structure

The completion and testing of major TSSW subsystems described in Section 2 are defined as level 2 milestones. Teams plan their work towards each test by defining a series of level 3 milestones. Teams may define level 4 milestones for their own use.
You can see the list of current milestones in Appendix E.
Resources to achieve the milestones throughout the duration of construction have been allocated by means of being loaded into the PMCS. Each top level WBS within TSSW (per Table 2) is divided into some tens of planning packages, each of which addresses some part of the TSSW baseline design with a clearly defined scope, deliverable, resource cost, and end date.

As the due date for work approaches, the actions required to complete each planning package— and hence meet the associated milestone *cycles* —must be defined in detail. This detail is defined in the TSSW Developer Guide[5].

[5] https://tssw-developer.lsst.io/

5 Products

The products of TSSW are the artifacts, systems, and services which will be used to operate LSST as an observatory. These are mapped fairly closely to the items listed in Appendix A. Each TSSW product is being developed to satisfy one or more of the requirements placed upon the TSSW subsystem.

6 Roles in Telescope and Site Software

This section describes the responsibilities associated with the roles shown in Section 3.

6.1 Director for Software (Deputy Project Manager)

The Director for Software is responsible for the efficient coordination of all LSST activities and responsibilities assigned to the Data Management Subsystem and Telescope and Site software. The Director for Software has the responsibility of establishing the organization, resources, and work assignments to provide TSSW solutions. The Director for Software serves as the software representative in the LSST Project Office and in that role is responsible for presenting software initiative status and submitting new software initiatives to be considered for approval. Ultimately, the Director for Software, in conjunction with his/her peer Project Managers (Telescope and Site, Camera), is responsible for delivering an integrated LSST system. The Director for Software reports to the LSST Project Manager. Specific responsibilities include:

- Manage the overall Software
- Define scope and request funding for Software
- Develop and implement the software project management and control process, including earned value management
- Approve the software Work Breakdown Structure (WBS), budgets and resource estimates
- Approve or execute as appropriate all software outsourcing contracts
- Convene and/or participate in all software reviews

6.2 Technical/Control Account Manager (T/CAM) – Software Manager

The Technical/Control Account Manager has managerial and financial responsibility for the engineering teams within TSSW. The T/CAM is responsible for a specific set of WBS elements and reports activities as required, including providing input for monthly status reports. The T/CAM is responsible for integrating TSSW's agile planning process with the LSST Project Management and Control System (PMCS). Specific responsibilities are to: Develop, resource load, and maintain the plan for executing the Telescope and Site construction project within the scope of their WBS, as well as to

- Synchronize the construction schedule with development in WBS elements managed by other T/CAMs
- Maintain the budget for their WBS and ensuring that all work undertaken is charged to the correct accounts
- Work with the Telescope and Site Project Controller (see Section 6.4) to ensure that all plans and milestones are captured in the LSST Project Controls system
- Perform day-to-day management of staff within their WBS
- Assist Director for Software in developing the TSSW plan
- Synchronize the TSSW plan, managed as per Section 4.3, with the LSST PMCS
- Ensure that the plan is kept up-to-date and milestones are properly tracked
- Create reports, Gantt charts and figures as requested by the Director for Software
- Serves as a member of the TSCCB.

6.3 Product Owner

A product owner is responsible for signing off on the functional requirements, the quality, prioritization and acceptance of a particular software product. The product owner works with the Customers (often a T&S CAM) and Stakeholders (end users such as the commissioning/operations teams) to ensure proper functionality is being delivered and determining the priority of feature implementation. The product owner then acts as the single point of communication from the stakeholders/customers to the software developers, translating the functionalities/requirements into user stories, reviewing acceptance test criteria, details for implementation and technical expertise.

The Telescope & Site software can be divided into three product categories (SAL, OCS and TCS), each with multiple components. Each product category is overseen by the Software Architect and each component has a Product Owner. The T&S subsystem scientist is the Product Owner of the final product as well as the integration of all the different areas.

The list of products and areas are as follows:

- Middleware (SAL), EFD
- Observatory Control System:
    - OCS executer, Scripting, Observing modes
    - Environment Awareness System: EAS
    - Visualization: LOVE
    - Scheduler
- Telescope Control System:
    - Calibration: in dome calibration and AuxTel
    - Active Optics System: M1M3, M2, Hexapods/Rotator – Alignment System
    - Observing tools: TMA, Pointing, Guider, Dome
    - Generic Camera CSC

Each product owner shall:

- Participate in bi-weekly sprint planning meetings, defining and reviewing each sprint
- Participate in the product owners bi-weekly meeting
- Gather the relevant stakeholders of each area and coordinate necessary meetings
- Define long term milestones (in collaboration with non-software CAMs when applicable)
- Sign off on the prioritization and descoping of functionality
- Work as a group, with T&S Software Manager, to assign personnel/resources appropriately
- Responsible for choosing the appropriate Working Group (as defined in section 3.2) in collaboration with the T&S Software Manager, T&S Subsystem Scientist, and T&S Project Manager.

The Product Owners will organize the necessary meetings with the relevant stakeholders to ensure buy-in on all of these issues. If there is a problem with agreements this may be escalated to the TSSW Manager and the T&S Subsystem Scientist. If the problem persists, this may be escalated to the Director of Software.

6.4 Telescope and Site Project Controller/Scheduler

The Project Controller is responsible for integrating TSSW's agile planning process with the

LSST Project Management and Control System (PMCS). Specific responsibilities include:

- Assist T/CAMs in developing the TSSW plan
- Synchronize the TSSW plan, managed as per Section 4.3, with the LSST PMCS
- Ensure that the plan is kept up-to-date and milestones are properly tracked
- Create reports, Gantt charts and figures as requested by the Software Deputy Project Manager.

6.5 Software Architect

The Software Architect has the responsibility for the design of the TSSW Software Architecture and for maintaining adherence to the agreed architecture. Specific responsibilities include:

- Define the overall architecture of the system and ensuring that all products integrate to form a coherent whole
- Identify appropriate technologies to be used in specific areas of subsystem development
- Minimize the exposure of TSSW to volatile external dependencies

The Architect will work closely with the Telescope and Site Systems Engineer and the Quality Assurance Lead to ensure that processes are in place for tracing requirements to the codebase and providing hooks to ensure that requirement verification is possible.

6.6 Lead Developer

The Lead Developer serves as owner for the major sections of the TSSW System (Test, Integration, Development, Architecture, etc.).

In addition, the Lead Developer provides technical expertise to the TSSW team. The Lead Developer will also work with LSST Project Systems Engineering on coordinating and executing Commissioning Activities Planning (CAP). The Lead Developer will select and advocate appropriate software engineering techniques and choose the technologies which are used within the codebase

They work with the T/CAM who has managerial responsibility for their product to define the overall construction plan and the detailed cycle plan for TSSW.

6.7 Integration Lead

The Integration Lead has the responsibility for integrating software applications, both within the TSSW engineering team and with external software providers (i.e. Camera). The Integration Lead works with the LSST Project Systems Engineering team to prioritize and execute Early Integration testing. Report activities as required, including providing input for monthly status reports.

- Develop and maintain the TSSW Simulation Cluster for testing reliability
- Define the contents of releases, in conjunction with the product owners, the TS Subsystem Scientist, and the technical managers
- Manage the software releases to the Simulation Cluster
- Works with LSST Project System Engineering on coordinating and executing Commissioning Activities Planning (CAP).
- Work with LSST AIV Management to coordinate TSSW deliveries to Telescope Integration process
- Work with technical managers to coordinate releases to match CAP and AIV testing schedules

6.8 Quality Assurance Lead

The Quality Assurance Lead has the responsibility for adherence to testing process for the TSSW engineering team. The Quality Assurance Lead works with the Lead Developer to prioritize development activities to address bug reports from testing. The Quality Assurance Lead also reports activities as required, including providing input for monthly status reports.

- Develop and maintain the integration tests between components and/or subsystems
- Develop and maintain the TSSW Release Policy as a change controlled document
- Manage the software release process and its compliance with documented policy
- Ensure that each release is accompanied by an appropriate documentation pack, including user manuals, test specifications and reports, and release notes
- Work with technical managers to coordinate bug fixes and maintenance of long-term support releases

6.9 Scrum Master

The Scrum Master has the responsibility for adherence to the agile development process for the TSSW engineering team. The Scrum Master works with the Product Owners and the T/CAM to maintain and react to priority changes in the development schedule. Report activities as required, including providing input for monthly status reports.

- Work with the relevant Telescope and Site Science Leads and Product Owners to develop a detailed plan for each cycle and sprint as required
- Create reports, charts and other metrics as requested by the T/CAM
- Perform the role of “Sprint-Lead” during agile development which include sprint task maintenance.

6.10 Senior Software Engineer

The Senior Software Engineers serve as mentors and provide technical expertise for the subsystem currently assigned (Auxiliary Telescope, M1M3, etc.).

Senior Software Engineers work with the Engineering Leaders who have responsibility for their subsystem to define and verify the overall completeness of the subsystem.

Senior Software Engineers develop quality subsystem software that meets the Product Owners supplied requirements. Senior Software Engineers develop Unit Tests and document the software produced for the subsystem.

6.11 Software Engineers

Software Engineers work with the Senior Software Engineers to develop quality subsystem software that meets the Product Owners supplied requirements. Software Engineers develop Unit Tests and document the software produced for the subsystem.

7 Development Process

Product development should follow a “Best Practices” approach to product generation. The TSSW team will follow the practices outlined in LTS-224[6] “Control Software Development Best Practices”. Some of the implementation of those practices are outlined in this document.

[6] https://docushare.lsst.org/docushare/dsweb/Get/LTS-224/LTS-224%20Control%20Software%20Development%20Best%20Practices%20V2.1.pdf

An agile process[7] is followed in TSSW. We have adopted a cyclical approach to software development, with major integration and operations activities as delivery points. TSSW will use JIRA as a tool to facilitate the agile process. Every three months, we define/refine a set of "epics", which correspond to major pieces of work to be undertaken during the cycle.
During the development cycle, all code is kept under continuous integration[8] (CI). Code versioning is managed in a Git[9] repository and is made available using an open source license. The release process of the code is defined using a modified version of the GitFlow workflow, detailed in section 3.5.1.

7.1 JIRA set-up

The JIRA tool that is used to facilitate the agile process. The workflow to be used is detailed in Figure 10.

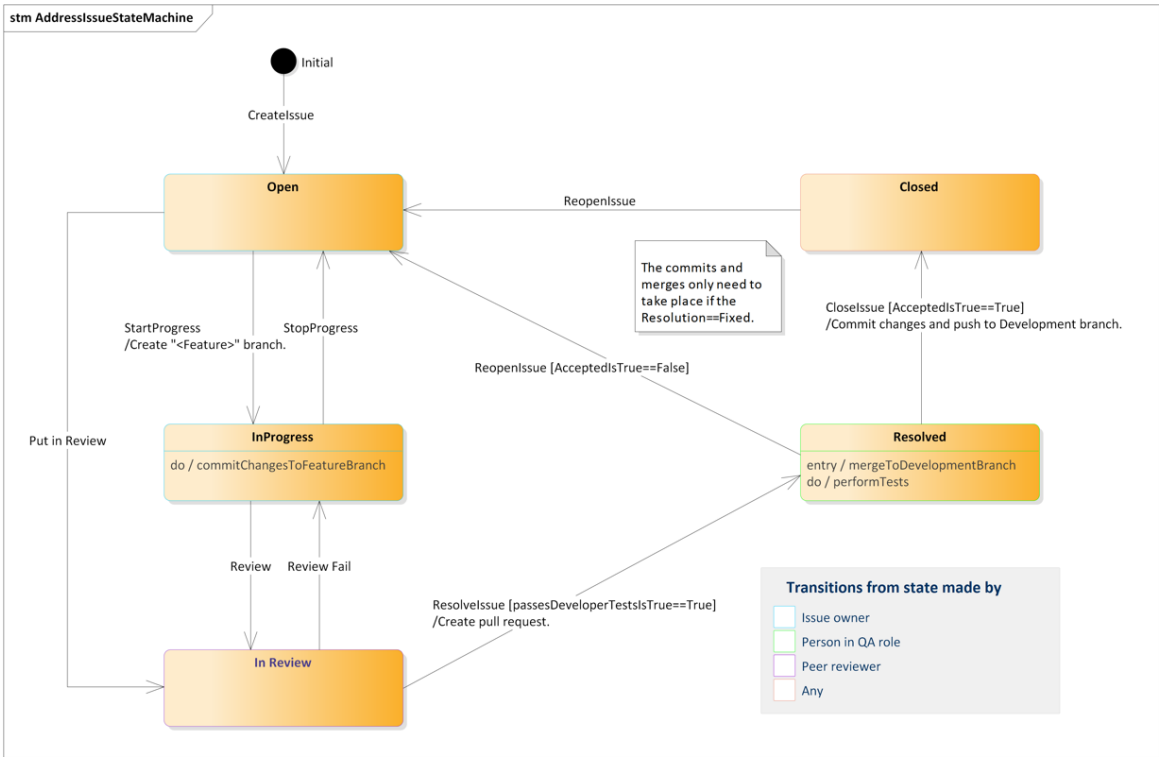

Figure 10: JIRA Workflow

[7] https://en.wikipedia.org/wiki/Agile_software_development
[8] Currently using the Jenkins tool; https://jenkins.io
[9] Currently using GitHub (https://github.com)

Description of the JIRA workflow is detailed in Appendix D.

7.2 Continuous Integration

Jenkins is the chosen Continuous Integration platform. Each application should have a build project in Jenkins. These applications should then have a dedicated build server.

Each build should run the unit tests. If the unit tests pass, the application should generate the deployable package and feed into the deployment system. Puppet[10] is the Deployment application.

The packaging system is still being investigated, but is leaning towards using RPMs, in general. For pure Python only applications, setuptools is a fine solution. LabVIEW may require another solution as well. This area is quite flexible, as the only real constraint is that it must be compatible with Puppet.

Appendices:

A. Components Listing

https://ts-xml.lsst.io/v/tickets-dm-20234/index.html

B. WBS and products

WBS Title	WBS no.	Charge no.	Charge no. Title
Management	1.04C.15.1	KLM41501A	T&S Software Management
Architecture	1.04C.15.1	KLM41501A	
Main Telescope	1.04C.15.2	KLM41502A	T&S Main Telescope Software
Pointing contract (remaining)	1.04C.15.2	KLM41502A	
AuxTel	1.04C.15.3	KLM41503A	T&S Auxiliary Tel Software
EFD/Middleware	1.04C.15.4	KLM41504A	Software Infrastructure
QA	1.04C.15.5	KLM41505A	T&S Software QA
Build system	1.04C.15.6	KLM41506A	T&S Software Build
Contracts INRIA	1.04C.15.7	KLM41507A	LSST Operator Vis Env (LOVE)
Contract INRIA 2	1.04C.15.7	KLM41507A	
OCS/TCS hardware	1.04C.15.8	KLM41508A	T&S Computing Hardware

[10] https://puppet.com/

Integration/Test (inc AIT)	1.04C.15.9	KLM41509A	T&S Software Int and Test

C. RFC Process

A larger change is proposed through a "Request for Comments" (RFC) that is managed through Confluence, since TSSW developers are familiar with the tool. By default, the proposer of an RFC is expected to be responsible for handling any effects of the change. If they propose an API change, they are agreeing that they will fix any breakage in other TSSW code; if they are proposing a change to the style guide, they will make the change. When an RFC is submitted it enters the Proposed state, the proposal is mailed to the TSSW developer list, and an announcement is made on the main TSSW Slack channel. Each RFC has a defined due date by which a decision is to be made. For changes that are expected to have no real impact outside of a single package, a due date of 3 days is acceptable. For changes that might result in significant debate, for example adding a new package to be supported by TSSW or changing an API used by many packages, the proposer is advised to give at least a week and possibly two weeks for debate. Anyone can comment on the RFC, and when consensus is reached the RFC can be Adopted by the proposer. If consensus could not be reached the proposer has the option of withdrawing the RFC completely, adding more time, or flagging the RFC to the Change Control Board for more formal decision making. On adoption, an RFC must be associated with actual work by filing tickets in TSSW Jira and connecting them with an "is triggering" relationship. The Implemented state is recognized by scanning all the Adopted RFCs and seeing that all triggered work has been completed.

D. JIRA Workflow description

Please refer to Figure 10 in regards to the description below:

- Initial/Triage
  - Create the Task, Bug or Improvement.
  - The initial assignee houd triage the ticket.
    - Ensure it is assigned correctly (Assignee, Component , etc)
      - Ensure the ticket is assigned to an actual, currently employed at LSST, person.
      - Each ticket is assigned a JIRA Component. If an Assignee is not designated, the JIRA Component will determine the assignee.
        - The component to assignee designation can be found here.
    - Ensure the Priority is set correctly (Ticket should not have the "Undefined" priority).

        - A Sprint can be chosen at creation (preferably by the person doing the work for this ticket). Otherwise the ticket is automatically placed in the Backlog. If the ticket is in Backlog, once the sprint is known, the ticket should be updated with the current sprint.
    - Once work begins, move the ticket to In Progress
- In Progress
    - Create the feature branch in the git repo and link the branch to the JIRA component.
        - No active development is ever done on the Master or Develop branches.
    - Document the requirements (via Requirements Template)
    - Write the code.
    - Write the unit tests.
    - Update release Notes & pertinent doc strings
    - When complete, move to In Review.
        - Complete meaning:
            - Add a link of the commit to the ticket.
            - Unit tests exist, have been successfully run and results have been added to the ticket or commit.
            - Add a link to the completed release notes
- In Review
    - Once the code is complete and all unit tests are passing, initiate a pull request on the develop branch and assign it to the Reviewer(s).
    - The Reviewer ensures
        - Code is complete and understandable.
        - Unit tests are passing.
        - Documentation is done, including a reference to the lsst.io site
        - All Requirements, as defined in the ticket Description, are met.
    - If findings occur
        - Updates Jira with findings
        - Sends back to developer
    - If no findings occur
        - Moves Jira ticket to review complete with approval/minor changes.
- Reviewed
    - The Developer then merges the pull-request (see Merge Process, below).
    - Moves ticket to Resolved.
- Resolved
    - This is the purview of QA.
    - QA does another spot review, to ensure the requirements are spelled out and properly met, all documentation is provided and the unit tests are passing.

    - This is also when QA works on the higher level tests (Functional, Integration, etc).
    - Once this is Complete, QA moves to Closed.
- Closed
    - The ticket is complete.
        - Feature was successfully implemented.
        - Feature was de-scoped; proper explanation provided.
        - Ticket was not implemented for some other reason; proper explanation provided.

E. Currently Defined Milestones (as of June 2019)

T	Key	Summary	Labels	Status
	DM-19395	Support M1M3 Cell w/Surrogate Mirror on TMA Testing	L2Milestone	TO DO
	DM-19394	Support M1M3 cell and surrogate mirror test with coating chamber	L2Milestone	TO DO
	DM-19396	Support for M2 Hexapod installation and test on TMA	L2Milestone	TO DO
	DM-19381	MainTel Calibration System Integration Tests in Chile	L2Milestone	TO DO
	DM-19369	M2 Functionality Check with CSC Controller + SAL - pt. 2	L2Milestone	TO DO
	DM-19368	M2 Functionality Check with CSC Controller	L2Milestone	TO DO
	DM-19380	AuxTel 1st Light Support	L2Milestone	TO DO
	DM-19379	Spectrograph On-Sky Support	L2Milestone	TO DO
	DM-19378	Auxtel Calibration System Integration Tests in Chile	L2Milestone	TO DO
	DM-19365	AuxTel & (Closed) Dome Pointing Control Test	L2Milestone	TO DO
	DM-19426	Support EAS instrument Site Testing	L3Milestone	TO DO
	DM-19413	MTWEP & MTOFC SW Int in Tucson	L3Milestone	TO DO
	DM-19406	AuxTel & Dome Pointing Control Test in Tucson	L3Milestone	TO DO
	DM-19415	Unpack & Install M2 controller in Summit computer room	L3Milestone	TO DO
	DM-19407	AuxTel Pneumatics & Pointing Configuration Integration w/ Hardware in Chile	L3Milestone	TO DO
	DM-19423	Support M1M3 Thermal Testing on TMA in Chile	L3Milestone	TO DO
	DM-19416	M2 Functionality Check with CSC Controller + SAL - pt. 1	L3Milestone	TO DO
	DM-19417	Auxtel Calibration System Integration Tests in Tucson	L3Milestone	TO DO
	DM-19418	Spectrograph On-Sky Integration	L3Milestone	TO DO
	DM-19419	AuxTel On-Sky w/ Spectrograph + Scheduler Support	L3Milestone	TO DO
	DM-19420	AuxTel 1st Light Integration	L3Milestone	TO DO
	DM-19421	MainTel Calibration System Integration Tests in Tucson	L3Milestone	TO DO
	DM-19422	Testing of M1M3 New Features w/ Hardware in Chile	L3Milestone	TO DO
	DM-19414	TMA Test Support	L3Milestone	TO DO
	DM-19425	Support M1M3 installation on TMA	L3Milestone	TO DO
	DM-19408	AuxTel Mount Control & Pointing Integration w/ Hardware in Chile	L3Milestone	TO DO
	DM-19427	Support M1M3 Thermal Testing on TMA in Chile	L3Milestone	TO DO
	DM-19434	Test ATHexapod Algorithm w/ Hardware	L4Milestone	REVIEWED
	DM-19437	MTDome Planning & Management	L4Milestone	TO DO
	DM-19431	AuxTel Mount Control & Pointing Integration in Tucson	L4Milestone	TO DO
	DM-19401	Testing of Weather Station SW	L4Milestone	TO DO
	DM-19402	Testing of DIMM SW	L4Milestone	TO DO
	DM-19429	Prepare Tucson Integration & Test Environment	L4Milestone	TO DO
	DM-19432	AuxTel Pneumatics & Pointing Configuration Integration in Tucson	L4Milestone	TO DO
	DM-19433	Test ATDome Algorithm w/ Hardware	L4Milestone	TO DO
	DM-19435	Test ATPneumatics Algorithm w/ Hardware	L4Milestone	TO DO
	DM-19436	Testing of M1M3 Mirror Thermal Control SW	L4Milestone	TO DO
	DM-19438	AuxTel On-Sky w/ Spectrograph + Scheduler Integration	L4Milestone	TO DO
	DM-19439	Testing of M1M3 New Features w/ Simulator in Tucson	L4Milestone	TO DO

F. Current Primavera Schedule (as of June 2019)