The LSST Scheduler from design to construction

Francisco Delgado*[a], Michael A. Reuter[a]

[a]LSST, 950 N Cherry Ave, Tucson, AZ USA

ABSTRACT

The Large Synoptic Survey Telescope (LSST) will be a highly robotic facility, demanding a very high efficiency during its operation. To achieve this, the LSST Scheduler has been envisioned as an autonomous software component of the Observatory Control System (OCS), that selects the sequence of targets in real time. The Scheduler will drive the survey using optimization of a dynamic cost function of more than 200 parameters. Multiple science programs produce thousands of candidate targets for each observation, and multiple telemetry measurements are received to evaluate the external and the internal conditions of the observatory. The design of the LSST Scheduler started early in the project supported by Model Based Systems Engineering, detailed prototyping and scientific validation of the survey capabilities required. In order to build such a critical component, an agile development path in incremental releases is presented, integrated to the development plan of the Operations Simulator (OpSim) to allow constant testing, integration and validation in a simulated OCS environment. The final product is a Scheduler that is also capable of running 2000 times faster than real time in simulation mode for survey studies and scientific validation during commissioning and operations.

Keywords: LSST, OCS, survey, scheduler

1. INTRODUCTION

The Large Synoptic Survey Telescope (LSST)[1] is controlled by a hierarchical architecture. The control system is organized in subsystems, and the high control entity that orchestrates these subsystems during operations is the Observatory Control System (OCS)[2]. The Scheduler is the OCS component that produces the targets for the observatory, implementing in fact the survey. Due to the variety of science goals and the operational constraints of the LSST observatory, the Scheduler was envisioned as a fully automatic and dynamic component, that according to the current conditions of the observatory and the environment, the past history of observations, and the multiple goals from the science programs defined in the survey, determines the next target at real time.

The need for such a Scheduler for the scientific success of the LSST was identified early on the project. In addition, the scale of the survey and the fact that the mission is about multiple sciences simultaneously, made the Scheduler a high risk component. In order to mitigate this risk, prototypes of the Scheduler were developed since the beginning of the R&D phase in the project, working in a highly detailed simulation environment. The result is a detailed design for the Scheduler[3], and a construction plan as a deliverable from the Telescope and Site team[4], that incorporates the simulation aspect for constant validation in coordination with Systems Engineering and the Simulations team.

2. PROTOTYPE HISTORY

Since early in the project, it became clear that current scheduling approaches in other astronomy observatories were not designed to deal with the operational concepts of the LSST and its philosophy of a single survey that serves many science cases. Space-based telescopes have elaborate schedulers, designed to execute detailed observing blocks, more constrained in pointing but free from unforeseen changes in external conditions, and where interruptions are handled by operators. Unlike for space-based telescopes, observing efficiency for a given target in ground-based facilities varies greatly with time due to atmospheric absorption. The presence of weather, the short exposures in the LSST with the

*fdelgado@lsst.org

frequent re-pointings and the sheer number of possible targets require that a scheduler must be able to re-plan autonomously and constantly while keeping the balance in the goals of all the science programs.

In order to explore this concept, a tool was created, the Operations Simulator (OpSim). Version 1 (OpSim1), written in IDL, consisted of a set of rules and priorities representing the LSST mission, and tested the ability of automatically respond to unscheduled interruptions. The algorithm was based on the computation of the demand in time for each target position on sky based on the requested observations by multiple science cases, and execute the demand depending on the telescope, the constraints and the environment. The result is a simulation of the visited positions with their conditions like airmass, seeing, etc., which was later used to evaluate the level of compliance with each science case.

OpSim version version 2 (OpSim2)[5] elaborated this concept in more detail and accuracy. It was written in Python and the models for the observatory and the environment were more comprehensive and fully configurable. The science cases were programmed into sophisticated science proposals evaluating thousands of targets against dozens of constraints and ranking rules competing simultaneously for the observation time of the simulated telescope. A target is defined as a field filter combination, and a visit is defined as a sequence of consecutive exposures for a single target, being a visit the observation unit of the survey. The current baseline specifies a visit as 2 exposures of 15 seconds. A database with the observation history captured all the simulated details for each visit, allowing the development of analysis tools to evaluate the survey performance and to estimate the capabilities of the observatory as designed. The major tool for this version was Simulation Survey Tools for Analysis and Reporting (SSTAR), which served assisting in the analysis of hundreds of simulations. All these runs with OpSim2 and SSTAR were key in proving that a design of a field of view of 3.5 degrees was needed to achieve the mission of the survey in 10 years. Also, it helped shaping the set of science proposals to achieve the high level goals of the Science Requirements Document (SRD)[6], and validating the telescope and dome specifications in terms of constraints and slew time. Simulations for 3 different candidate sites also supported the decision of selecting Cerro Pachón as the LSST site.

OpSim version 3 (OpSim3)[7] was a step forward, exploring more scheduling strategies and adding more fidelity in the telescope constraints, filter changer rate in the camera, the scheduled and unscheduled downtime, and more variations in the science proposals[8]. The scheduler components were better separated from the simulation components. The analysis tool was also upgraded, consisting now of the Metrics Analysis Framework (MAF)[9], provided the needed extended automation for parameter space exploration. The scheduling algorithms in OpSim3 also experimented with limited look-ahead capabilities, which demonstrated the potential and the challenges of using deterministic future knowledge about the target conditions to improve the survey efficiency.

All this effort gave the knowledge and the experience needed by the project to produce a solid design for the Scheduler and an organization between the teams to face the challenge of building a critical component of the control system of the observatory, that due to its combined set of goals, constraints and scale of the problem space, was a high risk element which was dramatically mitigated.

3. TWO CHALLENGES, ONE SCHEDULER

Another clear conclusion from the prototyping and simulation campaign is the need for an OpSim version during operations, in order to constantly assess the survey progress and to evaluate potential changes in priorities, parameters, system constraints, or even new science cases. In order to achieve a valid simulation of the Scheduler in operations, the operational version of OpSim must use the same scheduling algorithms. The number of factors affecting the final schedule of observations is so high, that the challenge of producing faithful simulations sharing only the algorithms or even libraries of code seems risky. On the other hand, developing a Scheduler exclusively in the OCS control environment would not exercise all the possibilities and the scientific validation before commissioning would be limited.

The decision was then to develop the OCS Scheduler in a framework that allows to run it directly in a simulation environment with the same code and the same interface used in the OCS control environment. To achieve this, the relevant functionality and telemetry from the OCS and the observatory subsystems have to be simulated in enough detail to reproduce the inputs needed by the Scheduler. This simulator is called Simulated Observatory Control System (SOCS)[10]. The combination of the Scheduler and the SOCS is then envisioned as OpSim version 4 (OpSim4).

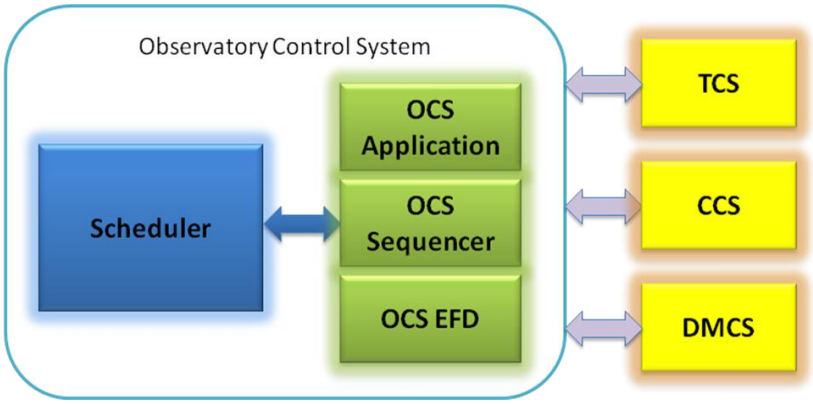

Figure 1. The Scheduler in the Observatory Control System context.

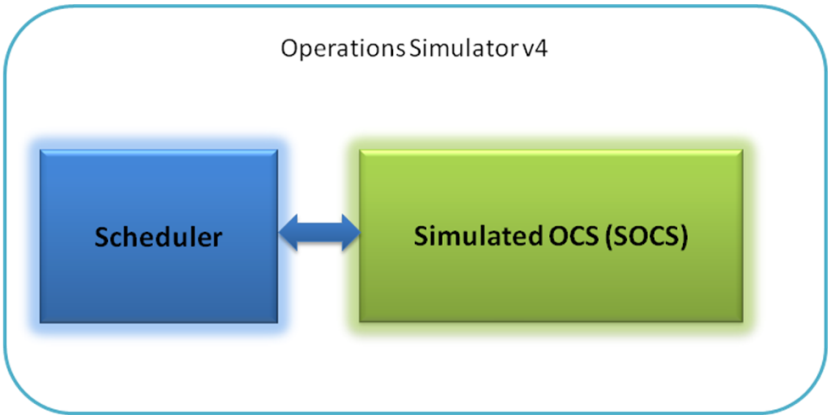

Figure 2. The Scheduler in the Operations Simulator context.

The approach of a single Scheduler implementation capable of running in 2 different contexts imposes some additional requirements. For the Scheduler, a complex set of algorithms and elaborated models previously planned to run in real time, now has to be capable with the same code implementation to run at faster simulations speed. The reference real time speed is the computation of the next visit during the current visit time, corresponding to a cycle of 1 visit in 39 seconds. However, other controllers in the observatory need a previous notification about the next visit or potential sequence of visits, which led to the requirement of computing a full night of visits during a visit time. This is an average of 700 visits in 39 seconds, corresponding to a cycle of 18 visits computed per second, 700 times faster than real time. Current full survey simulations with OpSim 3 typically takes 40 hours, which is equivalent to this speed, but the goal for the simulations in the future OpSim4 era is to achieve 2000 times faster than real time, simulating the full survey in less than 14 hours allowing a turnaround of simulation and automatic report of 24 hours. This speed imposes on the Scheduler the requirement of producing 50 visits per second interacting with SOCS, or 2000 times faster than real time.

For SOCS there is also an important implication of this strategy: the interface implementation. In OpSim3, the simulation code and models were in direct communications with the scheduling algorithms in one single python process. Now, the Scheduler is a standalone component in the OCS[11], with a publish/subscribe Data Distribution Service (DDS)[12] interface. SOCS must use this same interface in order to operate with the Scheduler. As simulations have to operate at high speeds, initial benchmarks were performed with the interface to validate the feasibility of the architecture, achieving cycles of 5000 visits per second in term of messages being exchanged between the initial versions of SOCS and the Scheduler with no targets being processed nor telemetry being simulated, just exercising the interface between both processes in the same computer, replicating the use case of OpSim4. This result demonstrated that the DDS interface will not be a bottleneck for fast simulations.

Another decision related to this approach is the continuation of the Python language in the coding of the Scheduler. While this software is closer to the control architecture, the uniformity with SOCS and the common models from the simulations team made Python the chosen language. This is also supported by the OCS team where a version of the DDS interface is also available for Python.

4. CONTROL ARCHITECTURE

The Scheduler in the OCS context obeys all the rules of the OCS components. The interface between the OCS and the other subsystems, Telescope Control System (TCS), Camera Control System (CCS) and Data Management Control System (DMCS) is based on the publish/subscribe Data Distribution Service (DDS). The OCS team develops a Software Abstraction Layer (SAL) that builds some additional services on top of DDS to make it available to the other subsystems. SAL is available in C++, Java, LabVIEW and Python. The OCS uses SAL to communicate its components internally as well, defining in this way the implementation of the Scheduler interface. With the Scheduler written in Python, SAL/Python is the flavor utilized.

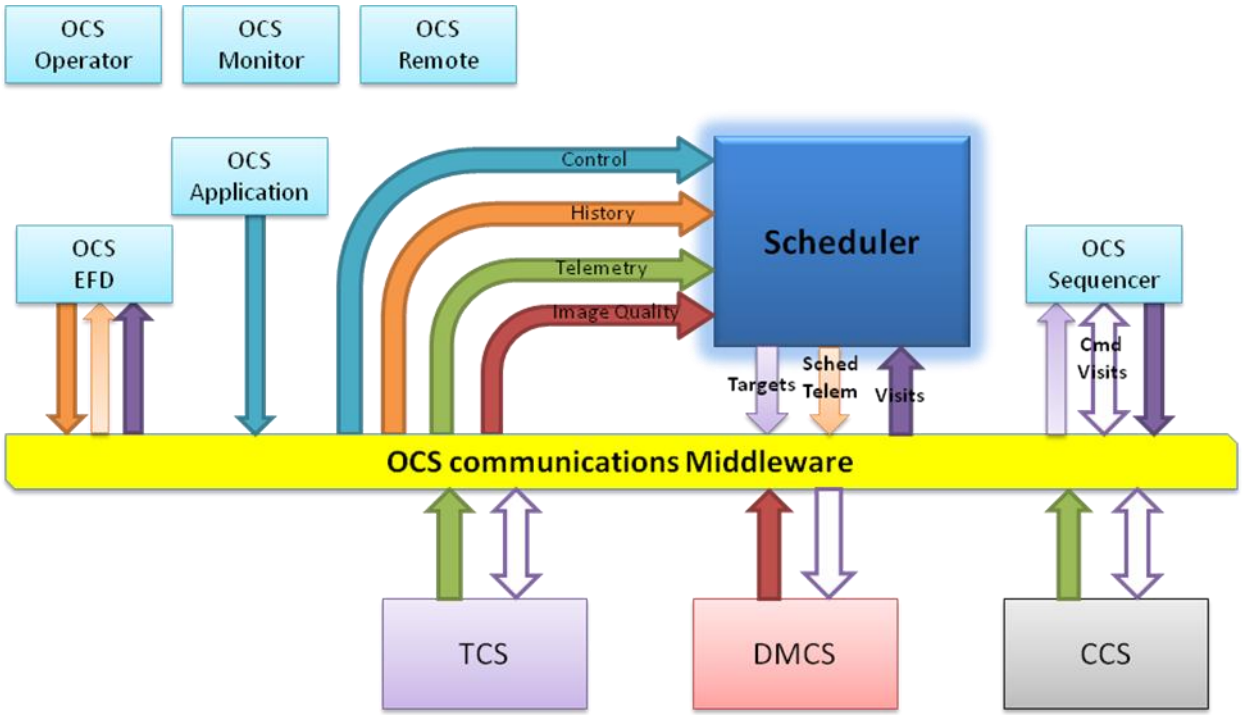

Figure 3. The Scheduler data flow in the Observatory Control System context.

The figure illustrates the context of the Scheduler as part of the OCS. The OCS Application controls the operational modes, the Engineering and Facility Database (EFD) provides the history of previous observations during start-up, TCS and CCS provide the telemetry for the current observatory conditions, TCS also provides telemetry about the environmental conditions, and DMCS provides the feedback with measured image properties from past observations. All this information is computed by the scheduling algorithms to produce the targets, main output from the Scheduler. the Sequencer executes those targets, and the actual observations are then notified back to the Scheduler to register the visits and adjust the priorities for the upcoming targets. Scheduler telemetry is also produced, with information regarding the decisions processes for monitoring the algorithms.

Below are the descriptions for the inputs and outputs of the Scheduler in this control context.

4.1 Control

This input comes from OCS Application, and includes all the commands that control the operational mode of the Scheduler, the configuration, and the handling of the downtime periods and degraded modes.

The configuration commands control the system parameters, scheduling parameters and the definition of the proposals to execute during the survey. Some of these can be changed dynamically at any time, others only during daytime, and others require a Scheduler restart.

4.2 History

During operations the Scheduler stores an annotated history of observations internally. However, in the case of a restart with a partial survey in progress, for example after introducing significant changes in the science proposals, a *warm start* function is activated in which the Scheduler queries the OCS EFD to obtain the full history of previous observations and with this list rebuild this internal annotated history which is fundamental to evaluate the partial progress of the science cases being pursued. This input represent the EFD feeding the history of the survey in a restart of the Scheduler.

4.3 Telemetry

The telemetry input includes internal and the external conditions of the observatory. The internal conditions are the positions and status of the telescope mount, rotator, dome, filter, used to evaluate the cost in time to new targets. The external conditions include but are not limited to the seeing, all sky transparency, measured sky background brightness, used to filter out targets that are outside the constraints defined.

4.4 Image Quality

The image quality feedback is a measure of seeing, sky brightness, m5 (the 5-sigma detection limit for point sources) among other parameters, coming from the image as taken and computed by some local component of Data Management. It is used to assess past observations and eventually re-evaluate the progress of the targets to modify the priorities of the re visits.

4.5 Visits

This is the control feedback of what is actually happening in the observatory regarding the observations. The Scheduler proposes targets constantly, but it's the OCS Sequencer who commands the observations; the actual visits performed by the observatory with the as taken telemetry and conditions is notified back to the Scheduler through this input in order to register the survey progress and to update the priorities for the future targets.

4.6 Targets

This is the main output from the Scheduler, and correspond to the sequence of targets, being updated constantly as observations go. The first target in this list is the next target to acquire, and this information is available one full visit ahead of time to give the Camera and Data Management enough time to prepare for that acquisition. The following targets are a prediction, but if conditions change to Scheduler re-evaluates this list.

4.7 Scheduler Telemetry

This is still an undefined output, and it is a placeholder to publish internal information that will help understanding the performance of the scheduling algorithms, like the alternate targets that were not selected and eventual intermediate ranking calculations.

5. SIMULATION ENVIRONMENT

In the context of the simulation environment, as part of OpSim4, the Scheduler works in exactly the same way with the same OCS Middleware interface. The design is such that the time signal controlling the status of the internal models and the day and night cycles comes from the telemetry and control signals. In this way, SOCS can feed an accelerated time signal altogether with the simulated telemetry and the Scheduler will operate in the accelerated simulation use case.

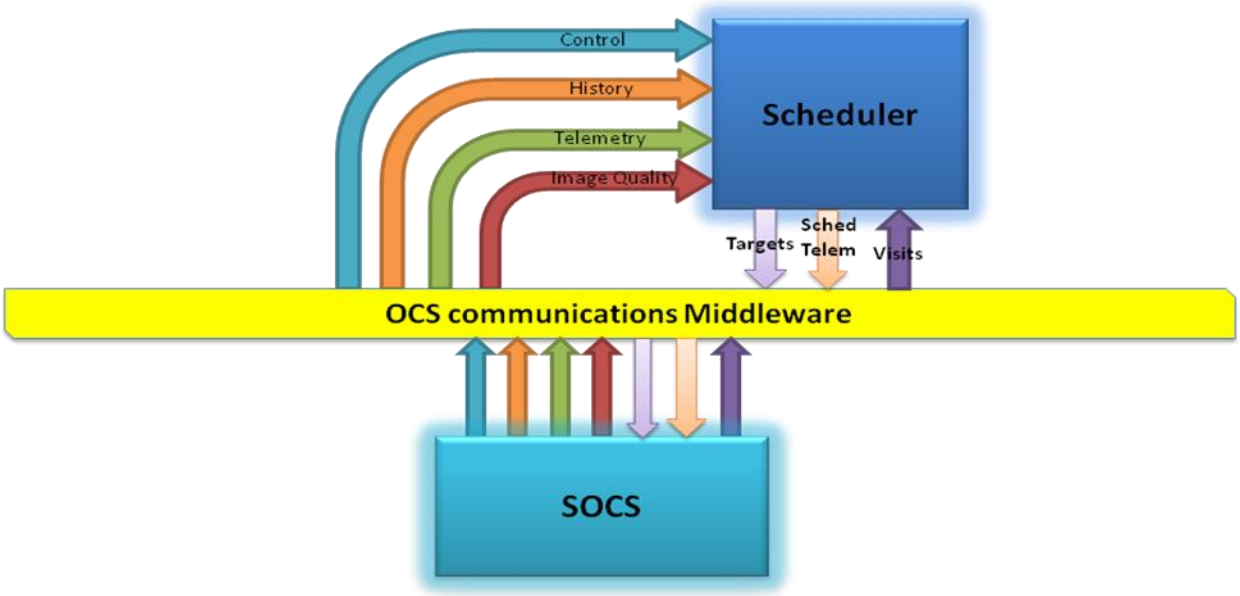

Figure 4. The Scheduler data flow in the Simulated Observatory Control System context.

In the figure 4, SOCS provides all the needed inputs to the Scheduler with simulated data and reacts with the observatory behavior to simulate operations and produce a survey database for further analysis.

6. DESIGN

The design for the Scheduler summarizes all the lessons learned during the prototyping and simulations phase of the project. It also complies with the standard interface framework for the OCS, and addresses with a modular design the speed requirements for the simulations use case. All the components are initially deployed inside a single process, with a design that allows for further parallelization of time consuming tasks. The architecture also makes possible to separate components of the Scheduler in different processes and utilize DDS as a messaging mechanism between the processes.

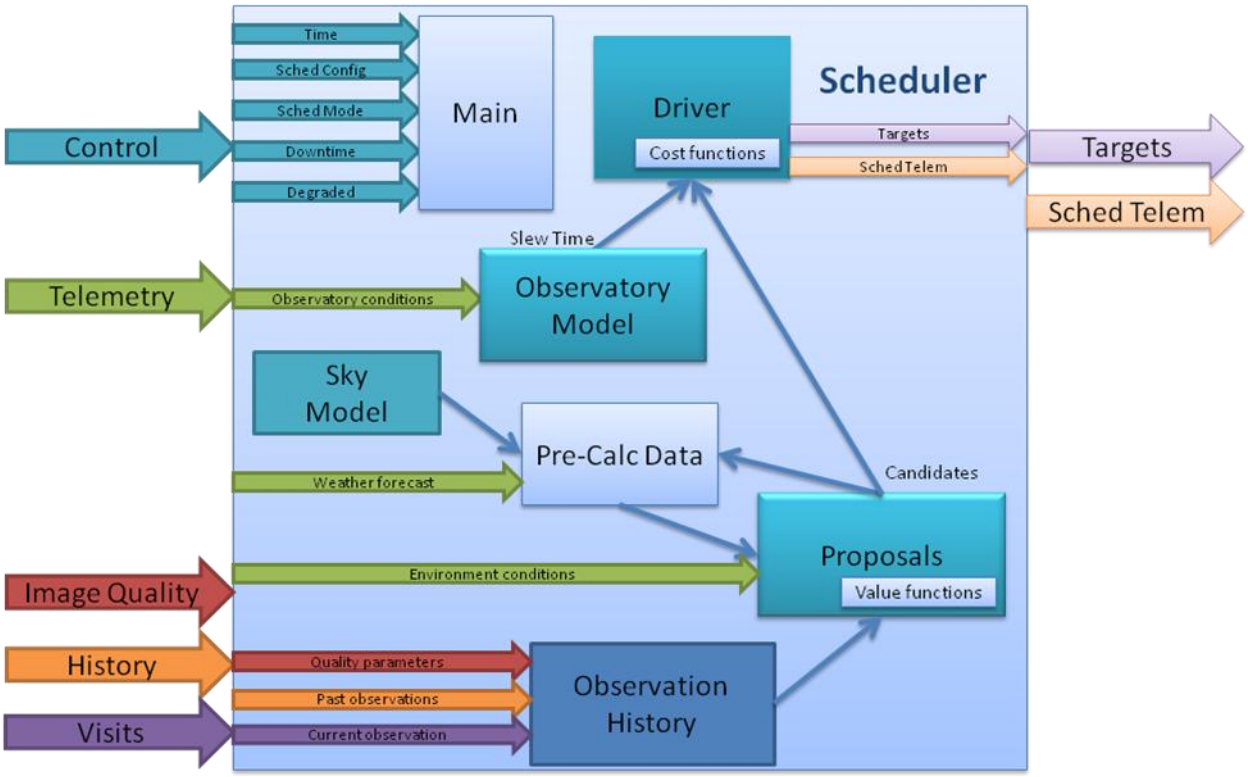

Figure 5. The Internal Block Diagram for the Scheduler with basic data flows.

6.1 Main

The component ***Main*** is first of all the entry point of the Scheduler code. It instantiates and initializes the framework including the logger and the SAL, and starts the ***Driver*** with default configuration. Secondly, ***Main*** is the implementation of the interface, including the command processor and the state machine that follows the standard defined for every OCS component. Every DDS message received or transmitted by the Scheduler is handled here, where the translation between the DDS topics and the internal data structures is performed. Lastly, this component controls the operation mode of the Scheduler, according to the control commands received from the **OCS Application**. Also, new configuration can also be received, and depending on the operation mode this new configuration is transferred to the ***Driver***.

6.2 Driver

The ***Driver*** is the component that actually conducts the survey. At start-up, it instantiates and configures all of the remaining components. It keeps track of the night and day cycles triggering the corresponding actions, maintains the state of ***Observatory Model*** aligned to the telemetry received, and orchestrates the ***Proposals*** to obtain the targets and rank their priorities. The cost functions are implemented in this component, where information about the targets' values coming from the ***Proposals*** is combined to the predicted slew time from the ***Observatory Model*** to build the configurable ranking function. Currently, the only cost considered is the slew time, but nothing prevents the inclusion of other considerations given a suitable model for those.

The ***Driver*** also implements the high level behavior of the Scheduler, everything that goes beyond the individual ***Proposals*** and that requires coordination with other components, such as the filter swapping during new moon. In order to implement this feature, the ***Driver*** follows the moon phase and when the configurable dark period is reached, it evaluates the relative progress per filter across the ***Proposals***, and decides which of the unmountable filters will be swapped from the camera for the **u** filter during new moon. This desired operation is then communicated to the **OCS**

Application at day time and executed by the operations crew. At the end of the dark period the reverse operation is requested by the Scheduler.

Dynamic configuration is also possible when new parameters are transferred from ***Main*** (after receiving a configuration command at the interface). The actions performed by the ***Driver*** to apply new parameters could be as simple as to update scheduling parameters, or more elaborately, like restarting the ***Observatory Model*** with new kinematic parameters, or starting a new ***Proposal***.

6.3 Proposals

The ***Proposals*** block in the diagram is the representation of the multiple instances of science cases configured in the Scheduler. There is a generic parent class called Proposal, with 3 specializations for Scripting, Area Distribution and Time Distribution. Any number of proposals can be running simultaneously in the Scheduler, with no communications between them. The ***Driver*** is the one in charge of preparing the common input data for them and handling their outputs.

Each Proposal is in charge of suggesting targets for the next visit. It receives the internal observatory state, the external environmental conditions, and after analyzing the previous history of observations delivers a valued list of targets. This process is repeated for each visit, providing the required dynamic nature of the Scheduler, adapting the list of targets to the variant external conditions and to the result of the competition among the multiple Proposals.

A Proposal implements a set of constraints to select the candidate targets at any given moment, and a value function to prioritize these targets. The specific value function depends on the class of Proposal, and the constraints are particular to each instance. Each instance is configured to implement a specific science case, and the most important constraints to select the targets are listed below:

- Sky region
- Airmass limit
- Hour angle limits
- Sky brightness limits per filter
- Sky transparency
- Seeing limit

Below are the descriptions of the three Proposal specializations.

6.3.1 Scripting proposal

This class of Proposal is configured with a fixed sequence targets that are delivered one by one in the pre-configured order.

6.3.2 Area Distribution proposal

The Area Distribution Proposal implements an even distribution of visits in a defined sky-region. The main parameter is the number of goal visits per filter for all the fields in the region, and the value function depends on the relative progress of each target to the overall advancement in the proposal. This simple approach proves to be effective in achieving the desired distribution, however the time domain aspect is strongly dependant on the conditions and targets availability, because it is not enforced in the value functions.

A variant of this proposal is going to be created in the future, when the look-ahead capability will be incorporated. In this look-ahead variation, the value function will consider the future availability of that field-filter given a defined rolling time window. The expected improvements are better efficiency in the area coverage and an improved distribution of observations during the survey time for these proposals.

6.3.3 Time Distribution Proposal

The Time Distribution Proposal implements a sequence of observations for each field in the defined sky-region. The time intervals between these observations are enforced with value functions following an algorithm based on time windows. This is the most complex class of proposal, due to the many variations for the sequence of observations:

- A sequence can be composed of multiple subsequences ranking simultaneously different filters at different intervals.
- A sequence can include a nested subsequence, where the main sequence controls the intervals to restart the subsequence.
- A sequence can be a special deep-drilling case, in which instead of scheduling simple visits for a field, a series of back-to-back visits in different filters is scheduled.

This set of algorithms has proven to be very effective in obtaining the desired time distribution for the visits in this type of proposal. The internal competition of targets in the proposal with time windows at different stages achieves a high level of completion.

A variant of this proposal with look-ahead capabilities will be incorporated in the future, where the value functions to organize the competing sequences will consider the availability of the targets in the following intervals. The expected improvement is a higher efficiency in the completion of sequences by avoiding interruptions due to unavailable targets in the look-ahead window.

6.4 Observatory Model

The Observatory Model is a kinematic simulation of all the moving axes and actions involved in the positioning of the observatory from one target to the next. It is a sophisticated and fully configurable simulation, which is also utilized in SOCS to actually produce the simulated telemetry during the observations. In the Scheduler context, the Observatory Model is used to estimate the slew delay time from the current position to each candidate target to evaluate the cost and produce the ranking function.

The simulated elements are:

- Mount Azimuth position with cable wrap
- Mount Altitude position
- Rotator position
- Dome Azimuth position
- Dome Altitude shutter position
- Active Optics corrections delay
- Camera filter change time
- Camera filter change rates limit
- Camera readout time

The axes are simulated with a second order kinematic model, constant acceleration and deceleration, in order to make the code fast to evaluate the millions of candidate slews in a night. The code is modular enough to expand it to a jitter limited model for increased fidelity if needed.

All the kinematic parameters, limits and times are configurable, including the definition of the actions that can be performed simultaneously and the ones that require a sequence.

6.5 Sky Model

The Sky Model component incorporates the sky brightness model by Yoachim et al.[13], and provides the background surface brightness for the fields in every filter for the given time. This model can't be replaced by eventually available telemetry, due to the pre-calculations needed by the scheduling algorithms in the Proposals, plus the look-ahead capabilities that require deterministic information in the rolling time window such as the expected sky brightness for a target at a specific point in the future.

This component also provides the position of the Sun for handling the night/day cycles and twilight, the Moon phase to activate the dark period observation with different filters and the Moon position to determine targets too close to the Moon to observe.

6.6 Pre-Calc Data

The pre-calculated data component is the container of deterministic information about the targets in a rolling look-ahead window. It is implemented by a SQLite database and in memory data structures. It fulfills 4 important objectives:

- Storage of non-variant data across the survey, for example, the field coordinates table.
- Storage and access of common data to all the Proposals, to avoid duplication of effort and improve efficiency. For example, current alt-az coordinates for fields and sky brightness.
- Storage and access of look-ahead data. Deterministic attributes of the known targets during a configurable rolling time window that can be used by the look-ahead features in the Proposals.
- For survey simulations, storage of pre-calculated data that does not change between simulations, to speed up the process.

6.7 Observation History

The Observation History is the component that stores the record of the visits performed by the observatory, annotated with the relevant information known specifically by the Scheduler and its science Proposals in particular, such as which proposal requested each observation, what is the sequence number if the visit is part of a Time Distribution Proposal, pairs counting, deep-drilling identification, etc.

The data in this component is built as the survey progresses, but it can also be built from scratch in the case of a warm-start, where the Scheduler is restarted with a new set of science Proposals and the history of previous observations is queried from the OCS Engineering & Facility Database, and that sequence of past visits is re-run against the Proposals to account for eventual partial progress.

Another important feature in this component is the ability to receive image quality feedback from DM about a past observation. The parameters sent by DM about a particular image are stored in the metadata of the Observation History for the corresponding visit, and the corresponding science Proposal is notified in order to re-evaluate that target and eventually adjust its value for future observations.

7. CONSTRUCTION PLAN AND STATUS

The construction plan for the Scheduler was laid out considering the opportunities that the project produced after the design and development phase. First, a very detailed set of requirements for the Scheduler, with a well understood set of features and mechanisms to implement the survey utilizing the designed observatory. Second, the existence of a prototype that reduced the risk in terms of evaluating the feasibility of the challenge and the assessment of the development cost. Last, the existence of a consolidated simulations team in Systems Engineering, with a construction plan for the Operations Simulator version 4, that will constantly integrate, test and validate the Scheduler in a Simulated OCS environment (SOCS).

The plan consists of 12 incremental releases, in coordination with an equivalent 12 incremental releases of SOCS. Each new release of the Scheduler will incorporate a pre-established set of new features that will be evaluated using SOCS. There are 4 types of evaluations for each release. First, the Scheduler code includes unit tests, which validates the behavior of the code in the isolated Scheduler context. Second, the integration test is when the Scheduler is utilized by SOCS to simulate survey operations, which validates that the code is performing as expected, implementing the desired functionality. Third, the scientific validation is when the Scheduler is exercised by the simulations team to analyze in detail the survey performance of the new algorithms. Last, each release is the integration to the OCS in the control environment.

The releases for the Scheduler in the plan are as follows:

- Scheduler Release v0.1
  - Code Framework
  - DDS-SAL/Python interface
- Scheduler Release v0.2
  - Time handling
  - Observatory Model
  - Scripted Targets Proposals
- Scheduler Release v0.3
  - Sky Model
  - Survey Driver
  - Area Distribution Proposals
- Scheduler Release v1.0
  - Filter swaps
  - Downtime
  - Time Distribution Proposals
- Scheduler Release v1.1
  - Look ahead for area distribution
- Scheduler Release v1.2
  - Look ahead for time distribution
- Scheduler Release v1.3
  - Performance enhancements
- Scheduler Release v1.4
  - Warm start
  - Image quality feedback
- Scheduler Release v1.5
  - Dithering
  - Spatial weather telemetry
- Scheduler Release v2.0
  - Predicted schedule

- Scheduler Release v2.1

  Weather forecast handling

- Scheduler Release v2.2

  Additional scheduling optimizations

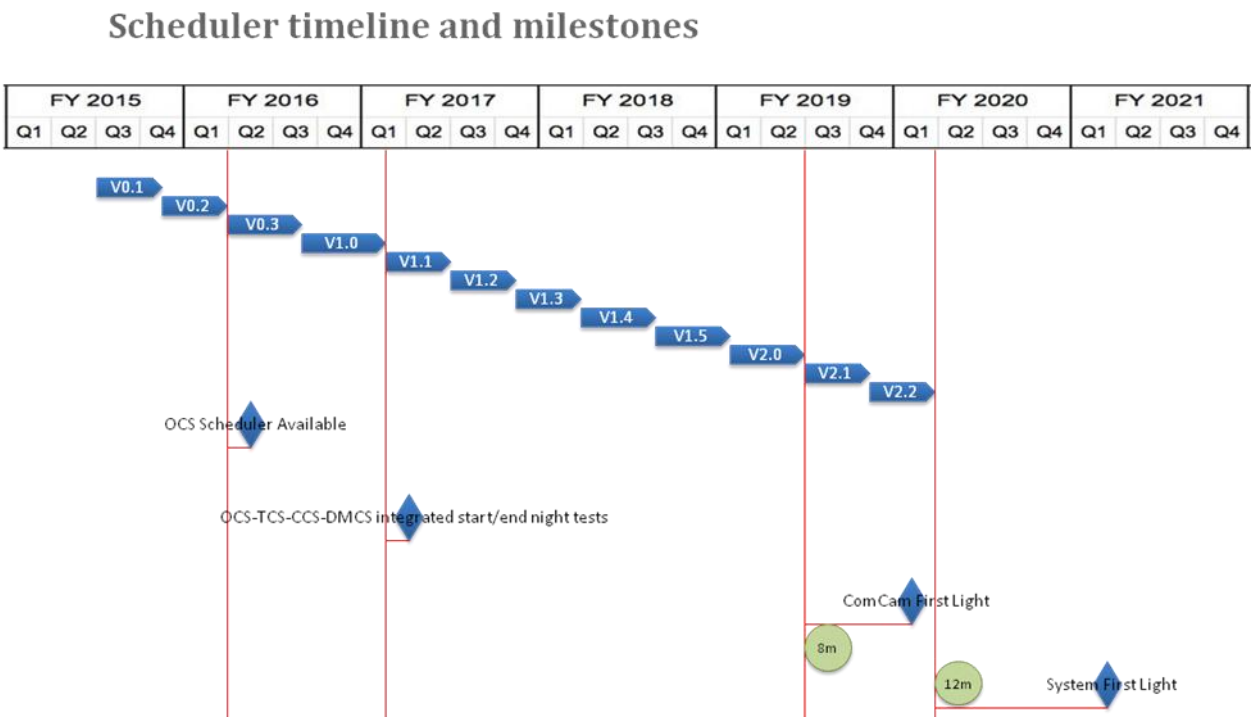

Figure 6. The Scheduler construction timeline.

The construction plan is on track, with the release v0.3 successfully tested. Version v1.0 is currently under development, and it will constitute an important milestone because it reproduces the features known in OpSim3. It will be validated by the simulations team to evaluate the performance in the baseline survey, and compared against the figures obtained in previous simulations. Version 2.0 will be available for the commissioning camera (ComCam) first light with a sufficient float of 8 months in the current plan, however version 1.5 will be good enough for conducting commissioning surveys. Version 2.2 with the complete set of features and optimizations in the current plan is expected to be released one year before the full system first light.

8. SOFTWARE ENGINEERING TOOLS

The analysis for the Scheduler design was done in SysML, Enterprise Architect, combining high level requirements flow down, behavior description, structure elements allocation and interfaces identification. The Systems Engineering model for the Scheduler is part of the integrated SysML model for the LSST[14], and regular updates are applied to this model as new information becomes available.

The detailed construction plan is described in JIRA, keeping track of individual tasks that mark the way to each release. The JIRA plan, releases, milestones and tasks are used periodically to report progress to the Project Management Control System (PMCS) and to organize the work of the development team.

The Scheduler code is kept in a Stash Git repository, following coding standards from Telescope & Site Software and Systems Engineering Simulations about templates, interfaces, code and unit tests. Continuous Integration (CI) is also implemented with Jenkins.

There is version control over the requirements document, high level OCS requirements and system level requirements documents are generated from the SysML models in Enterprise Architect, stored in Docushare. Also version controlled are the Scheduler code, the SOCS code, the simulation libraries, the system and survey parameters, and the baseline simulated 10 years survey. The simulation parameters for the current production in OpSim3 are stored in GitHub repository, and plans are in place to set a system for the soon to come OpSim4 simulations with the real Scheduler.

The construction plan is based in the knowledge about the science requirements and the system design available at that moment. In the probable case of change in requirements coming from the science or the engineering, the project has a formal system to handle those. A change request can be submitted to the Change Control Board (CCB), and in the Systems Engineering team the request is analyzed and actions can be triggered to evaluate the modification and its impact in the science, the budget or the delivery dates. Simulations can be requested to OpSim to support the studies, and if accepted the documentation and the plans will be updated, closing the loop in the JIRA projects and the PMCS.

9. TESTS AND VALIDATION

Integration and testing in the simulations environment is planned for each of the 12 releases with SOCS. These tests are focused on the expected behavior of the units in the Scheduler, and analyzing the simulated survey to validate that the algorithms are producing the expected results in terms of achieving the configured constraints and time domain distribution of the observations.

From version 1.0, which reproduces the capabilities explored in the prototype contained in OpSim3, scientific validation will be also performed. MAF and other analysis and reporting tools will be updated to assess that the generated survey is complying with the scientific goals of the LSST, and also that the parameter space can cover a wide variety of science cases to test the flexibility of the design in the Scheduler.

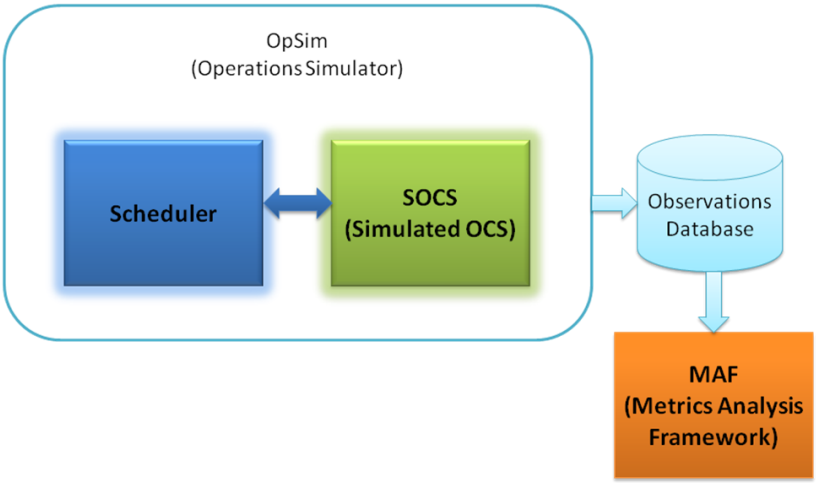

Figure 7. The Scheduler in the OpSim4 context for technical tests and scientific validation.

This analysis not only covers the validation of the scheduling algorithms and the impact on the science, it also is a powerful process to explore survey variations and parameters optimizations. One important deliverable of this process is the reference simulated survey[15], which is version controlled by Systems Engineering. This reference survey is updated when important changes are applied to the science proposals formulation or the system design.

10. SUMMARY

In this paper we presented the history of the Scheduler design since its initial prototype in simulations and the evolution towards a modular design capable of producing the targets for the observatory as part of the Observatory Control System, and also capable of producing full simulated surveys as part of the Operations Simulator. The design was exposed in both contexts, and the plan to produce the Scheduler have been laid out showing the coordination with the development plan for the Simulated OCS with regards to continuous integration, test and validation. The projected result is a software deliverable that will be capable of producing the survey during operations, produce simulations for studying alternate surveys and assess potential updates to the parameters of the science cases and the systems for constant optimization.

ACKNOWLEDGMENTS

Financial support for LSST comes from the National Science Foundation (NSF) through Cooperative Agreement No. 1258333, the Department of Energy (DOE) Office of Science under Contract No. DE-AC02-76SF00515, and private funding raised by the LSST Corporation. The NSF-funded LSST Project Office for construction was established as an operating center under management of the Association of Universities for Research in Astronomy (AURA). The DOE-funded effort to build the LSST camera is managed by the SLAC National Accelerator Laboratory (SLAC).

REFERENCES

[1] Kahn, S., "Final design of the Large Synoptic Survey Telescope," in [Ground-Based and Airborne Telescopes IV], Hall, H., Gilmozzi, R., and Marshall, H. K., eds., Proc. SPIE 9906-17, in press (2016).

[2] Daly, P., Schumacher, G., Delgado, F., and Mills, D., "LSST OCS status and plans," in [Software and Cyberinfrastructure for Astronomy IV], Chiozzi, G., Guzman, J, C., Proc. SPIE 9913-112, in press (2016).

[3] Delgado, F. and Schumacher, G., "The LSST OCS scheduler design," in [Observatory Operations: Strategies, Processes, and Systems V], Peck, A. B., Benn, C. R., and Seaman, R. L., eds., Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series 9149, 91490G (2014).

[4] Gressler, W., "LSST Telescope and Site Status," in [Ground-Based and Airborne Telescopes IV], Hall, H., Gilmozzi, R., and Marshall, H. K., eds., Proc. SPIE 9906-19, in press (2016).

[5] Delgado, F., Cook, K., Miller, M., Allsman, R., and Pierfederici, F., "LSST operation simulator implementation," in [Observatory Operations: Strategies, Processes, and Systems], Silva, D. R. and Doxsey, R. E., eds., Proc. SPIE 6270, 62701D (2006).

[6] Ivezic, Z., and the LSST Science Collaboration, "Large Synoptic Survey Telescope (LSST) Science Requirements Document," (2011) http://ls.st/LPM-17

[7] Delgado, F., Saha, A., Chandrasekharan, S., Cook, K., Petry, C., and Ridgway, S., "The LSST operations simulator," in [Modeling, Systems Engineering, and Project Management for Astronomy VI], Angeli, G. Z. and Dierickx, P., eds., Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series 9150, 15 (2014).

[8] Saha, A., Ridgway, S. T., Cook, K. H., Delgado, F., Chandrasekharan, S., Petry, C. E., and the Operations Simulator Group, "Advancing the LSST Operations Simulator," in [American Astronomical Society Meeting Abstracts #221], American Astronomical Society Meeting Abstracts 221, 247.03 (2013).

[9] Jones, R. L., Yoachim, P., Chandrasekharan, S., Connolly, A. J., Cook, K. H., Ivezic, Z., Krugho_, K. S., Petry, C., and Ridgway, S. T., "The LSST metrics analysis framework (MAF)," in [Observatory Operations: Strategies, Processes, and Systems V], Peck, A. B., Benn, C. R., and Seaman, R. L., eds., Proc. SPIE 9149, 91400B (2014).

[10] Reuter, M., Cook, K. H., Delgado, F., Petry, C. E., and Ridgway, S. T., "Simulating the LSST OCS for Conducting Survey Simulations Using the LSST Scheduler," in [Modeling, Systems Engineering, and Project Management for Astronomy VI], Angeli, G. Z. and Dierickx, P., eds., Proc. SPIE 9911-83, in press (2016).

[11] Lotz, P., Dubois-Felsmann, G. P., Lim, K., Mills, D., Johnson, T., Chandrasekharan, S., Schumacher, G., Delgado, F., Daly, P. N., Pietrowicz, S., Selvy, B. M., Sebag, J., Marshall, S., and Jenness, T., "LSST Control Software Component Design", in [Software and Cyberinfrastructure for Astronomy IV], Chiozzi, G., Guzman, J, C., eds., Proc. SPIE 9913-9, in press (2016).

[12] Mills, D., Schumacher, G., and Lotz, P., "LSST Communications middleware implementation", in [Ground-Based and Airborne Telescopes IV], Hall, H., Gilmozzi, R., and Marshall, H. K., eds., Proc. SPIE 9906-204, in press (2016).

[13] Yoachim, P., Coughlin, M., Angeli, G. Z., Claver, C., Connolly, A. J., Cook, K. H., Daniel, S., Ivezic, Z., Jones, R. L., Petry, C. E., Reuter, M. A., Stubbs, C. W., and Xin, B., "An Optical to IR Sky Brightness Model For the LSST", in [Observatory Operations: Strategies, Processes, and Systems VI], Peck, A. B., Seaman, R. L., and Benn, C. R., eds., Proc. SPIE 9910-48, in press (2016).

[14] Schumacher, G., and Delgado, F., "The Large Synoptic Survey Telescope OCS and TCS models," in [Modeling, Systems Engineering, and Project Management for Astronomy IV], Angeli, G. Z., Dierickx, P., eds., Proc. SPIE 7738, 77381E (2010)

[15] Cook, K. H., Delgado, F., Petry, C. E., Reuter, M. A., Ridgway, S. T., Jones, R. L., Yoachim, P., Connolly, A. J., and Ivezic, Z., "Exploring possible Large Synoptic Survey Telescope (LSST) surveys with the LSST operations simulator and delivering a reference simulated survey," in [Modeling, Systems Engineering, and Project Management for Astronomy VI], Angeli, G. Z. and Dierickx, P., eds., Proc. SPIE 9911-49, in press (2016).