The LSST operations simulator

Francisco Delgado, Abhijit Saha, Srinivasan Chandrasekharan, Kem Cook, Catherine Petry, et al.

Francisco Delgado, Abhijit Saha, Srinivasan Chandrasekharan, Kem Cook, Catherine Petry, Stephen Ridgway, "The LSST operations simulator," Proc. SPIE 9150, Modeling, Systems Engineering, and Project Management for Astronomy VI, 915015 (4 August 2014); doi: 10.1117/12.2056898

Event: SPIE Astronomical Telescopes + Instrumentation, 2014, Montréal, Quebec, Canada

The LSST operations simulator

Francisco Delgado*[a], Dr. Abhijit Saha [b], Srinivasan Chandrasekharan [b],
Dr. Kem Cook [c], Catherine Petry [d], Dr. Stephen Ridgway [b]
[a]Cerro Tololo Inter-American Observatory, Casilla 603 La Serena CHILE;
[b]National Optical Astronomy Observatory, 950 N Cherry Ave, Tucson AZ USA;
[c]Eureka Scientific, Berkeley CA USA;
[d]LSSTC, 933 N Cherry Ave, Tucson AZ USA

ABSTRACT

The Operations Simulator for the Large Synoptic Survey Telescope (LSST; http://www.lsst.org) allows the planning of LSST observations that obey explicit science driven observing specifications, patterns, schema, and priorities, while optimizing against the constraints placed by design-specific opto-mechanical system performance of the telescope facility, site specific conditions as well as additional scheduled and unscheduled downtime. It has a detailed model to simulate the external conditions with real weather history data from the site, a fully parameterized kinematic model for the internal conditions of the telescope, camera and dome, and serves as a prototype for an automatic scheduler for the real time survey operations with LSST. The Simulator is a critical tool that has been key since very early in the project, to help validate the design parameters of the observatory against the science requirements and the goals from specific science programs. A simulation run records the characteristics of all observations (e.g., epoch, sky position, seeing, sky brightness) in a MySQL database, which can be queried for any desired purpose. Derivative information digests of the observing history are made with an analysis package called Simulation Survey Tools for Analysis and Reporting (SSTAR). Merit functions and metrics have been designed to examine how suitable a specific simulation run is for several different science applications. Software to efficiently compare the efficacy of different survey strategies for a wide variety of science applications using such a growing set of metrics is under development. A recent restructuring of the code allows us to a) use "look-ahead" strategies that avoid cadence sequences that cannot be completed due to observing constraints; and b) examine alternate optimization strategies, so that the most efficient scheduling algorithm(s) can be identified and used: even few-percent efficiency gains will create substantive scientific opportunity. The enhanced simulator is being used to assess the feasibility of desired observing cadences, study the impact of changing science program priorities and assist with performance margin investigations of the LSST system.

Keywords: LSST, simulation, modeling, scheduling, survey

1. INTRODUCTION

The LSST survey is based on the assertion that an integrated observing strategy and campaign can serve a wide variety of science goals. The four pillars of LSST science specified in the Science Requirements Document (SRD) cover cosmology (especially dark matter and dark energy characterization); time-domain astronomy; Galactic structure and local volume investigation and exploration of the solar system, especially Near Earth Objects (NEOs) and Trans-Neptunian Objects (TNOs). Repeated imaging of the available sky will yield depth through “stacking” or “co-addition” of the data, as well as temporal information through the multiple epoch observations in each of the pass-bands which span a wide spectral range. This wide coverage of a host of scientific areas is only possible if individual and specific science goals can be met. This will happen only when the correct operating strategies are put in place to attain the necessary total depth in each pass-band, adequate temporal sampling, adequate range of parallax factors for the astrometry, and so on. How does one arrive at such an integrated observing strategy in the face of not only the cyclical

Modeling, Systems Engineering, and Project Management for Astronomy VI,
edited by George Z. Angeli, Philippe Dierickx, Proc. of SPIE Vol. 9150, 915015
© 2014 SPIE · CCC code: 0277-786X/14/$18 · doi: 10.1117/12.2056898

elements like diurnal, lunar and solar patterns, but also unpredictable stoppages due to weather, inadequate image quality, and so on?

The way forward is through simulation, using an adaptive algorithm that can respond to changing requirements and limitations. We can assert what the properties and limitations of the telescope and observatory will be; we know the statistical properties of the site where we are building – what seeing and weather patterns to expect as a function of season – and we know of what restrictions celestial patterns impose on what observations we can make. These define our boundaries, against which we test what set of observations we wish to obtain**.** We need a tool to adaptively design an observing strategy that will get us the integrated data set we desire. And finally, we want to engage this tool, once we have learned from it and fine-tuned its parameters, to conduct the actual survey for us, allowing room for necessary mid-course corrections and adjustments.

What is OpSim?

The Operations Simulator (OpSim) is a software tool that runs a survey simulation with given science driven desiderata; a software model of the telescope and its control system; and models of weather and other environmental variables. The output of such a simulation is an "observation history", which is a record of times, pointings and a record of associated environmental data and telescope activities throughout the simulated survey. This history can be examined to assess whether the simulated survey would be useful for any particular purpose or interest.

Such a tool must necessarily have several key elements:

- Mechanism(s) for specifying science driven desiderata
- A high fidelity and detailed model of the telescope and instrument, incorporating their limitations and operational overheads
- Reliable models for weather and other environmental attributes (including target availability in the sky at any given time), how they progress with time and season at the telescope site, and other situations that interrupt observing
- Optimization methods and algorithms that make operational decisions that maximally deliver the desiderata given the constraints of telescope, instrument and environment
- A set of tools to help interpret the observation histories: tools that help diagnose the mechanics of the logic delivering the desiderata, and tools that digest the history and report on aspects that are of interest to the prospective user community through figures of merit, metrics, and other representational forms.

Given satisfactory implementation of the above elements, scenarios with varying desiderata can be run to examine not only whether the survey can deliver its stated goals, but also to quantify margins in the survey performance, and how to best utilize it to accommodate additional scientific objectives. Further, if correctly configured, the same tool can also drive the real time survey. By the time of LSST survey operations, OpSim should contain within it the "knowledge" and means to deliver an optimal survey given a realizable set of desiderata. It is thus a project goal for the Operations Simulator Team to design, optimize, and deliver the scheduling algorithms necessary to drive the real time LSST survey.

This document provides an overview of the current status of the OpSim tool and the Operations Simulator Team's role in the LSST project. While it continues to be improved, OpSim is now sufficiently mature for generating simulated surveys that are useful for science planning and assessment. Its current architecture allows continuing improvement to all of the elements mentioned above, and facilitates mating this tool to the LSST telescope control system (TCS).

Why Build from Scratch?

There are scheduling tools for astronomy in existence, especially for space-based observatories. There are also tools employed by ground-based observatories for scheduling service or queued observations. Because many aspects of the LSST "mission" are operationally unique, especially in that it is an integrated survey catering to a multiplicity of science goals, it is not unexpected that some of the scheduling requirements are unique. Some key points worthy of note are as follows:

- Ground based observing is interrupted with much higher frequency than space missions. This makes carrying out a pre-determined timeline for observations difficult, and especially so when temporal links between observations are an important driver.
- Ground based service mode observing has historically been proposal-based, and schedule details often worked out by hand. Pan-STARRS[3] is the first project that can be thought of as a survey on a scale similar to LSST. Their implementation of global automated scheduling was not suitable for application to LSST, though they were interested in earlier incarnations of OpSim for that purpose. Pan-SSTARS exposures are much longer and fewer, and they appear to have used a queued observing model. Theirs is a collection of missions, not an integrated strategically cadenced, multi-purpose observing campaign.
- The pace and cadence of LSST survey observations (~1000 pointings per night) is too rapid and the set of telescope and instrumental constraints is too large to manage manually – even if it is to modify a pre-determined schedule in response to environmental changes. For instance, a particular science driven demand may be to observe field X in seeing better than S. This cannot be pre-scheduled successfully, since atmospheric conditions are unpredictable. Manual management of ~1000 pointings a night in these circumstances is not feasible.

It is clear that scheduling decisions must be made on all time scales. On the shortest scales, we may have to choose between multiple possibilities with differing science driven priorities, subject to cost functions driven by mechanical considerations such as the length of a slew. Recovering from a stoppage in the middle of the night requires an instantaneous decision for where to begin. On long time scales: over a 10-year survey period, the science horizons will change, and new desiderata may replace older ones. The survey desiderata, submitted as "proposals" to the simulator, with assigned priorities for each, are used to create instantaneous demand functions for each pointing element (field/filter combination) in the sky for any instance in time during the survey. This is tempered against the "cost" of making a particular observation, and a decision of what to observe at that instant can be made according to a suitable optimizing algorithm.

An over-arching design goal has been to make this tool as automated as possible: provide the desiderata, telescope model and environmental input, and let the rules of engagement in the optimization algorithms make the pointing decisions. To improve the outcome, recast the desiderata or alter the optimization rules, and run the simulation again. Learn to control and tune a simulated survey through experimentation, and implement the procedures that achieve the best performance during operations for the real time survey.

2. HISTORY

It became clear early on that scheduling programs used in other astronomy facilities were not designed to handle the operational challenges of LSST, with its "one survey serves many" philosophy. The more sophisticated schedulers available were designed for space-based facilities, which are typically far more constrained in pointing, where they are typically executing discrete observing blocks that are specified in detail within science programs. They do not have to manage changes in observing conditions (seeing, transparency, etc.), and recoveries from major interruptions are handled by human interaction. In contrast, the most pressing need for LSST -- where visits are of very short duration, revisits are frequent and numerous, and time domain requirements are at a premium -- is addressing the issue that re-commencing from unscheduled interruptions (weather) requires a re-plan of the entire timeline. The sheer number of possible pointings and the short timescales for such decisions require that the software be able to do such re-planning autonomously on all time scales, balancing the science requirements and priorities of the project.

Version-1 of OpSim was a demonstration that such a hands-off simulator, where a user provides a set of rules and priorities and the program is able to figure the response to unscheduled interruptions, while complying with a set of scheduling rules is feasible on the scale of LSST. The central idea was to set up and track the time dependent 'demand function' for each tessellated position on the sky based on the observations desired by the various science case 'clients', and respond to that demand modulo telescope, environmental, and programmatic constraints. The output of a simulation was the list of fields (or pointings) along with any qualifiers (such as seeing, airmass, etc.) at each epoch which were imaged over the entire duration of the survey. This set of visits could then be queried to examine the efficacy with which the observations served the purpose of any particular science case.

Version-2 of OpSim fleshed out this idea, and in particular, incorporated a model of the capabilities and constraints of the telescope as designed to available detail and accuracy. This version was written in Python, consistent with the 'open source' concept of the LSST project. There was continuous enhancement of Version-2, to specify science-driven observing constraints and cadence rules with increasing sophistication. The observation history records from the simulation runs were annotated by an increasing set of information pertaining to each pointing: a parallel effort to summarize the human digestible information from these records to assess usefulness for key science objectives was required, and a tool called SSTAR was created for this purpose. Experiments with OpSim have informed the project about science performance, and influenced telescope design and engineering studies: the results from an early version of OpSim influenced designing to a field of view to a 3.5 degree diameter so that coverage for the survey could be accommodated. *Using the high fidelity modeling of the telescope performance, and using actual weather patterns from CTIO over 10 years, Version-2 produces simulations that meet SRD formal design requirements.*

The currently released Version-3 is a major rewrite of the code with enhancements. It was motivated by three considerations:

- While Version-2 accomplishes the SRD requirements for number of visits and depth, it does not necessarily provide the optimal way in which observations should be carried out, nor does it optimize for the most desirable cadence that can be attained. Further investigation along these lines may provide additional margin needed for accomplishing additional LSST science objectives. In Version-2, all the decision-making of where to point to next is instantaneous: i.e., it responds with a "greedy algorithm" to the demand function of each potential field as it stands at the time of deciding the next observation, with no consideration of what this "move" implies in the future. This was a limitation that prevented implementing algorithms that can produce more efficient schedules, since any improvement over the greedy algorithm requires the ability to "look ahead" and examine whether a proposed visit to a target within a time window in the future is feasible or not. "Look ahead", most generally, refers to assessing the feasibility or the "opportunity space" available for future observations of a given field/filter that are required to complete a desired set of observations. A simple example is that in Version-2 there is nothing to prevent initiating a sequence whose future observations cannot possibly be completed (e.g., a revisit is required in 30 minutes, but dawn will break sooner than that). Or it may warn the code that it is behind on the expected cadence rate for field X, and soon it will set for the season, so it needs to boost the priority for observing that field. In other words, the infrastructure required to track feasibility in future time windows has been built into Version-3. This infrastructure will make it possible to implement and test alternative algorithms that we expect will drive the schedule to be more efficient, and thus increase observational margin.
- The end goal is to evolve to the Scheduler, which will drive the telescope during the survey. In real time operation, environmental data, like seeing, cloud cover, and sky brightness will be available, and should be used for scheduling decisions. In simulations, these stimuli are themselves simulated. The Version-3 design modularizes the code so that these simulated stimuli can be replaced with real input.
- The astronomical community, which has a large stake in the design of the LSST survey, has an interest in using the OpSim code to evaluate possible scheduling scenarios and their impact on science productivity. What changes in cadence, in support of a particular science objective, can be supported without adversely affecting other science objectives? The code must be portable, and tools to interpret the output of simulations in the context of the science cases must be supplied. The OpSim code will be used to run a variety of simulations with a wide range of parameters and evaluate their performance for specific science use cases. The results of these runs will provide benchmarks for LSST performance under various operational assumptions. These results are a way of communicating the capabilities of the LSST survey to the community: LSST Science Collaborations will have access to certain approved simulated surveys, and others will be available on request.

Version-3.0 reproduces the functionalities of the past versions, and is the platform for future enhancements, and the implementation of some of these are already in progress.

3. SIMULATOR ARCHITECTURE

The LSST Observatory Control System will feature the Scheduler as the main component. A prototype of this Scheduler is included in the OpSim, and this prototype will continue evolving to become the actual Scheduler. Given these plans and design choices, the Simulator architecture has to be built around the Scheduler prototype. The interface of this Scheduler prototype has to be identical to the interface of the Control System Scheduler, at least in the input/output definitions as understood at this stage of the project. Then, the Operations Simulator surrounds this Scheduler prototype with the needed Control System components and simulated environment.

First, the actual expected environment for the Scheduler has to be considered. This is, the surrounding OCS elements, the internal telemetry, and the external conditions. Figure 1 illustrates the operations environment for the Scheduler, its input and output in general:

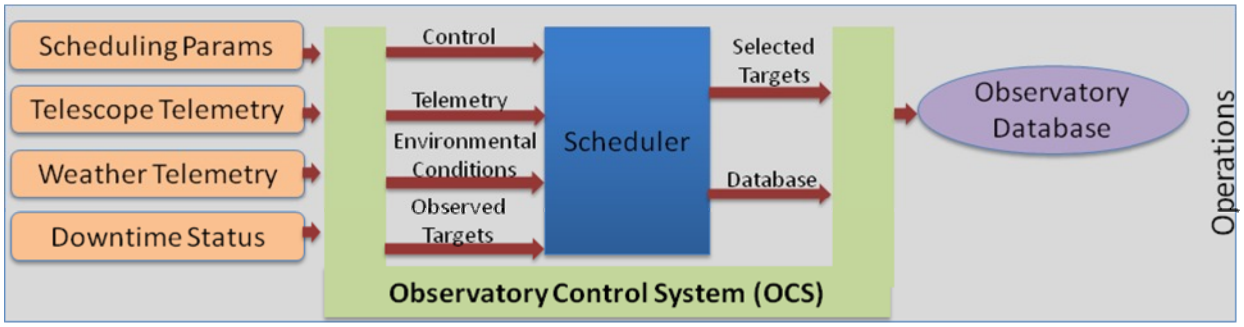


Figure 1. The control Scheduler, the blue box, inside the Observatory Control System environment.

Now, the Simulator has to provide the same harness by simulating the needed external conditions and the observatory telemetry. Figure 2 shows the context that the Operations Simulator has to provide for the Scheduler prototype:

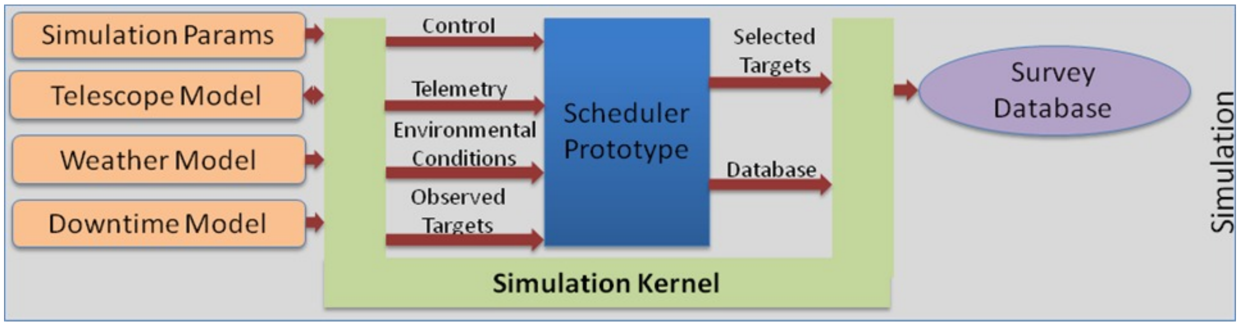


Figure 2. The Scheduler prototype, the blue box, inside the Operations Simulator environment.

The context analysis can be translated to the block definition diagram and the internal block diagram for the Operations Simulator architecture. The basic components are:

- **Simulation Kernel**: the control section of the simulated operations environment.
- **Telescope Model**: the observatory telemetry simulator.
- **Weather Model**: the external conditions telemetry simulator.
- **Downtime Model**: the downtime simulator.
- **Scheduler**: the scheduler prototype.
- **Survey Database**: the output of the simulation.

The chosen development environment was Linux and Python language. Python offers a very good blend of object-oriented capabilities, high productivity, rapid prototyping, complex and useful native data-types, easy integration with libraries from other languages, and full portability to Mac OS X. These characteristics make this interpreted language an attractive option for the LSST Operation Simulator.

The communications architecture for the LSST control software is a distributed paradigm, based on DDS, following the publish/subscribe protocol. The same architecture is envisioned internally for the Observatory Control System. Taking that future development into account, the architecture of the Operations Simulator is compatible to that approach; even it is currently implemented in an object oriented python single process environment, the interfaces of its components are designed to support an evolution towards a distributed implementation with message flows instead of just methods calls. Figure 3 illustrates the basic data flows for the Operations Simulator components.

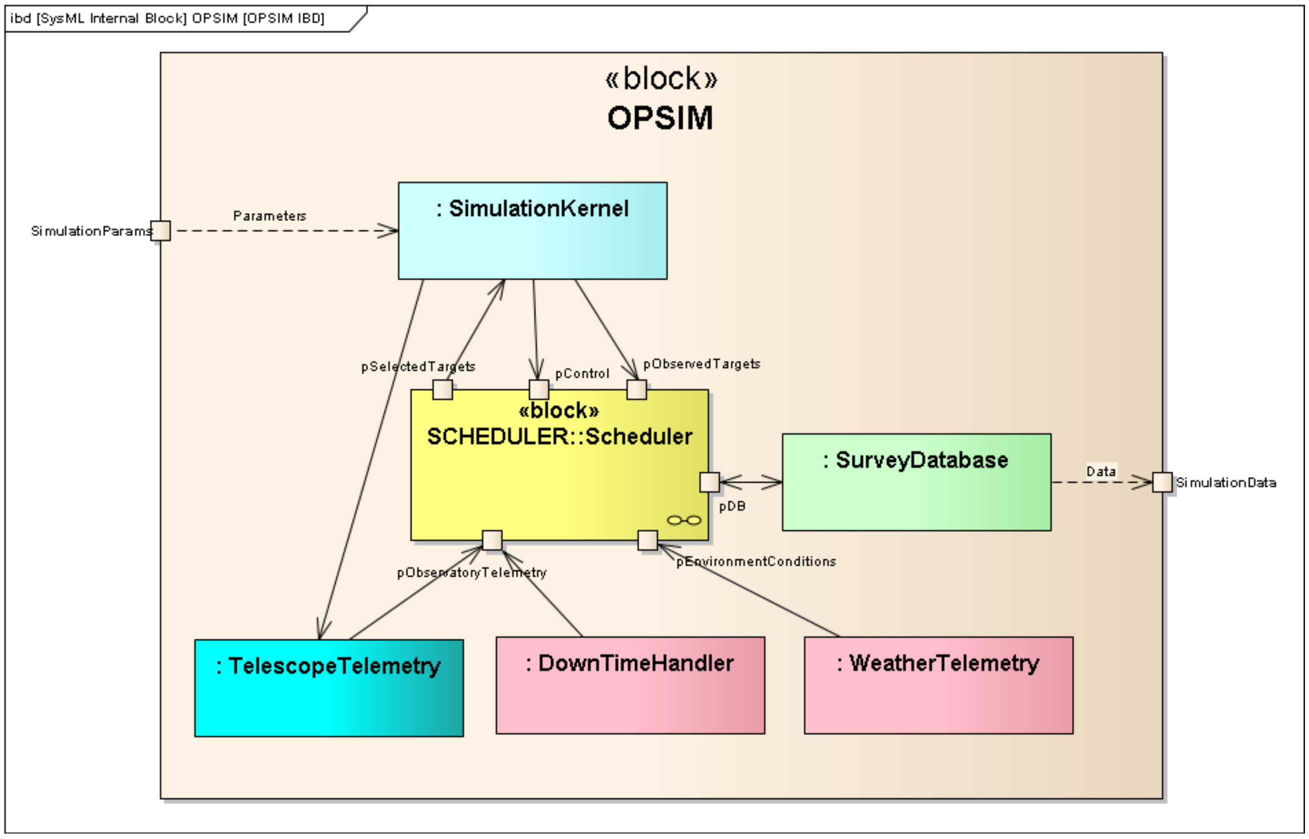


Figure 3. SysML internal definition diagram for the Operations Simulator.

The SimulationKernel is the entry point of the system; it reads the configuration parameters, initializes other objects and sets the simulation up. During the simulation run it keeps track of the simulated time and stores general information into the database.

To decide the action in every observing opportunity, the kernel asks the Scheduler for the next proposed target. To achieve this the Scheduler starts obtaining the weather and observatory status from the simulation blocks WeatherTelemetry and TelescopeTelemetry.

DowntimeHandler provides the availability of the observatory and SurveyDatabase stores the full detailed history of the survey, which is key information used by the Scheduler for the decision making process.

The heart of the operations is the Scheduler, the software component that considers the history of past observations and the current conditions, and selects the next target.

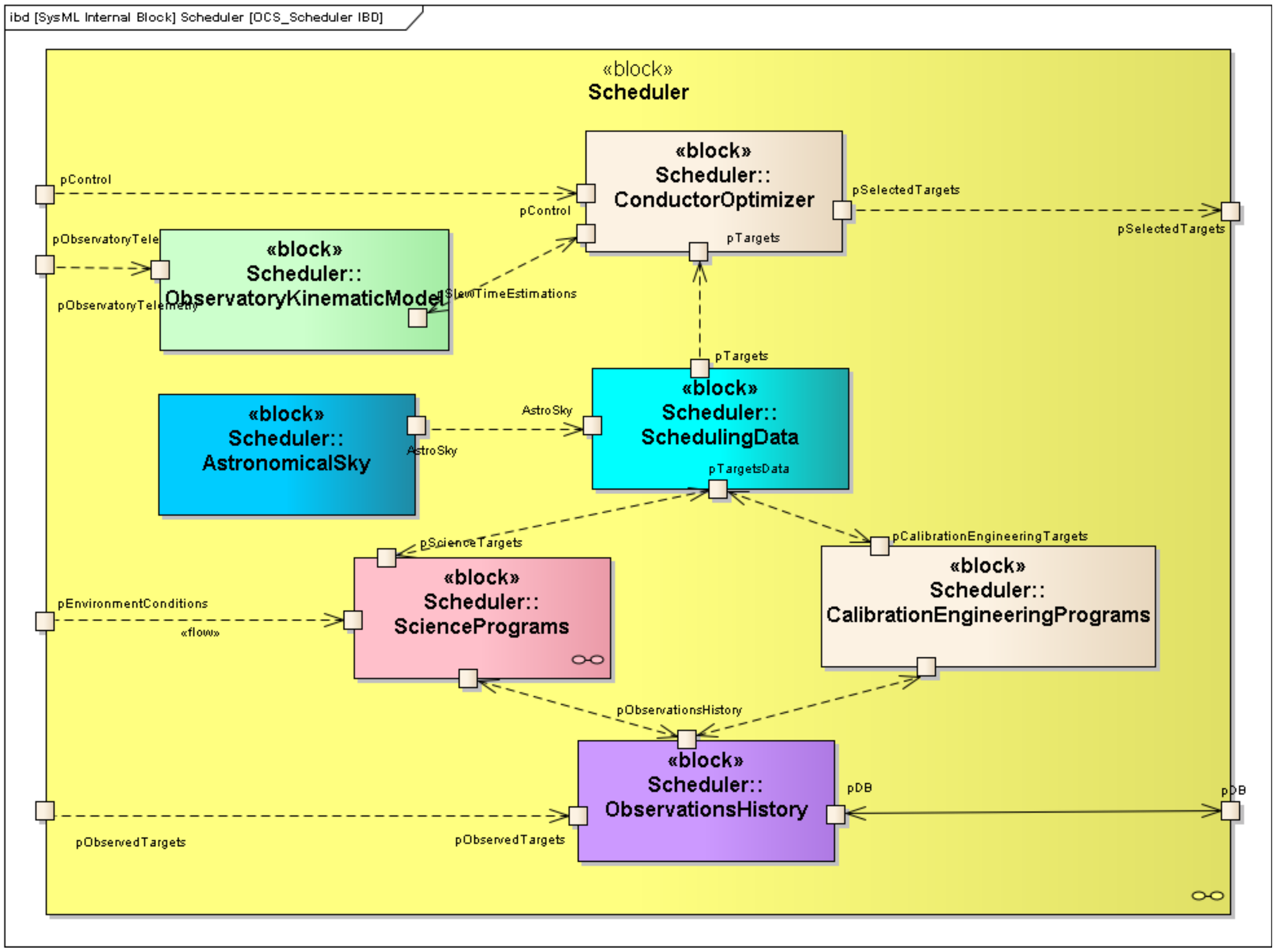

ibd [SysML Internal Block] Scheduler [OCS_Scheduler IBD]
«block» Scheduler
«block» Scheduler:: ConductorOptimizer
«block» Scheduler:: ObservatoryKinematicModel
«block» Scheduler:: AstronomicalSky
«block» Scheduler:: SchedulingData
«block» Scheduler:: SciencePrograms
«block» Scheduler:: CalibrationEngineeringPrograms
«block» Scheduler:: ObservationsHistory
pControl
pControl
pSelectedTargets
pSelectedTargets
pObservatoryTele
pObservatoryTelemetry
SlewTimeEstimations
pTargets
pTargets
AstroSky
AstroSky
pTargetsData
pScienceTargets
pCalibrationEngineeringTargets
pEnvironmentConditions
«flow»
pObservationsHistory
pDB
pDB
pObservedTargets
pObservedTargets

Figure 4. SysML internal definition diagram for the Scheduler.

Input: Configuration Files

There are several configuration files, basically one for each component of the simulator.

Main: a primary file containing the general setup, start time and duration of the simulation, site for the observatory and science proposals to activate.

Site: one file for each of the 3 sites originally considered, with the observatory coordinates and identification of the weather tables.

Telescope: the complete set of parameters for the telescope simulator, including hardware limits and slew parameters such as accelerations and speeds.

Proposals: one file for each of the survey programs, specifying the sky region, observations required for each field, cadence requirements if any, filter selection options including seeing limits and sky brightness limits, rank parameters, etc.

Scheduler: scheduling parameters such as the number of candidates requested of each proposal for every visit, slew time penalty and the number of observations to schedule in an observation block.

Output: Survey Database

MySQL was chosen for the database. There is a single database for the Simulator, where each simulation run is tagged by a session ID. This ID is used in all the tables that store the simulation details. Some of these tables are:

- Fields: a unique table with the fields of all the sky with the coordinates for each field center.
- Weather: a seeing table and a clouds table for each site.
- Configuration: the configuration parameters for the run.
- Time History: time events in the run such as day/night and twilight times.
- Observation History: all the visits performed in the simulation, with the target coordinates, sky conditions, weather, slew delay, filter used, rank, proposal of origin, etc.
- Sequence History: the final status of the sequences from the proposals with transient targets.
- Slew History: a set of tables with the telescope movements, delays and detailed slew activities for each visit.

Simulation Kernel

This module is the main driver of the simulation. It deals with the reading of the configuration files, setting up the simulation, initializing all the components according to the configuration parameters, and keeping track of the simulated time during the run and handling the components accordingly. The kernel calls the Scheduler and DownTimeHandler requesting the next target and a down-time event respectively, and selects the next action to perform. The sunset, sunrise, twilights and moon phases are obtained from the AstronomicalSky module inside the Scheduler, and these events trigger its operation.

Weather Telemetry

The weather module is actually a simple interface to the database tables that store the seeing and clouds data collected for the sites.

For the seeing data, the database table contains the seeing values in intervals of 5 minutes for 2 years. This data comes from seeing monitors at each of the sites. The weather module delivers then the seeing value for any time, taking the closest previous data and wrapping the 2 year sequence for longer simulations or start dates outside the range of the stored measurements.

The cloud table in the database contains a single numeric value for a measure of cloudiness (where 0.0 is clear, 0.5 is partial and 1.0 is cloudy), every 3 hours for 1 year. This data comes from historic records from Cerro Tololo for the same years. The weather module delivers for a given time the cloud value for the closest previous measurement, wrapping around the year.

Telescope Telemetry

The telescope telemetry simulator was designed to keep track of the position of the most important telescope components during the simulation, and more importantly, to calculate delay time during slews. It basically works by simulating the tracking of a target during an observation. It updates the state at the end of the exposure, and computes the slew time from that position to any given target provided by the Scheduler.

When a new observation is performed, the model slews to the new position updating the state, recording information into the database, and simulating the tracking movement so it gets ready for the next iteration. There is also a configurable parking position which is taken at down-times and at the end of the night.

There is a kinematic model for the dome, the mount and the rotator. There is also a model for delay vs. altitude-slew for the telescope optics corrections, open loop and closed loop. The filter change delay is modeled as well. The delay contribution from each of these components is calculated according to the initial and final telescope positions, taking into account the activities that can be performed in parallel and the ones that need to be serialized. This model is powerful because it calculates the slew time based on a detailed telescope model instead of considering only the distance on sky of the targets. In this way, the effect of physical design parameters on the surveys can be assessed.

All the movements of the simulated telescope are stored in database tables, including the coordinates for every axis before and after every exposure, maximum speeds reached, slew delay contribution for every component in the model, etc.

Downtime Handler

This component of the Simulator manages the defined scheduled and unscheduled downtime of the observatory. The behavior is fully configurable and during the simulation it keeps track of the availability of the observatory.

Scheduler

The Scheduler is the prototype of the Observatory Control System Scheduler for the LSST. The concept is to follow the same set of input, output and behavior so the OpSim serves as a test environment for the functionality of this key control component of the observatory. The Scheduler decides automatically the best target for each visit opportunity. Its normal cycle starts updating the atmospheric conditions for each field in the aggregated list of possible targets for the night. For this it uses the weather telemetry and astronomical sky modules. After that it calls to each active proposal requesting a list of suitable candidates for observation, passing along the updated atmospheric conditions of the fields to them. The Scheduler then builds a list of all the field-filter combinations received from the proposals, keeping the rank value provided by them and combining the targets that are suitable for multiple proposals adding their rank. This feature promotes the selection of targets that satisfy multiple surveys.

SchedulingData

The SchedulingData object concentrates all the target information, and it considers a future time window of pre-calculated information. This gives the possibility of activate **look ahead** behavior in the proposal ranking process or in the optimization process for selecting the best sequence of visits according to the particular merits to maximize.

Once this list is ready the telescope simulator is consulted about the slew time penalty of each candidate. The Scheduler then combines the rank value from the proposal with the slew delay from the telescope model using the appropriate parameters, and computes for each target an overall rank value. Depending on the value of some parameters, the user can choose to optimize for slew time or for a science goal, because there could be some candidates far away from the current position that have urgency at the moment, and a high slew penalty would give them lower priority compared to less urgent but closer alternatives. The top ranked field-filter from this final list is then selected for observation and proposed to the simulator kernel.

There is a special parameter for reusing the lists from the proposals for an observing block of a given number of observations, optimizing the execution time, so only the slew time is computed for every candidate on every visit. Every granted observation by the simulator kernel is notified to the scheduler and forwarded to each proposal, so they can update their history and goal achievements.

Astronomical Sky

This module implements the ephemeris for the sun and the moon, and a model for the sky brightness distribution from Krisciunas and Schaeffer (1991). There is a time and spatial grid cache feature for optimization purposes.

Observatory Kinematic Model

The Observatory Kinematic Model is another instance of the telescope model. It is used inside the Scheduler, and it is necessary for calculating the potential slew delay for each candidate observation during the scheduling analysis. This component has to be different in state from the external telescope telemetry simulator, and this fact is more obvious in the case of the real Scheduler in the Observatory Control System, where real telescope telemetry comes as input but potential slew delays are also needed for calculation the cost of the targets prior to the visits.

Science Proposals

The scientific proposals are the objects whose role is to propose targets to the Scheduler at each observation moment, giving a numeric rank to each candidate target according to their own agendas. Each proposal has its own scientific objective and individual characteristics, but all of them come from a parent class that implements the basic behavior and structure.

4. SCHEDULING VISITS

With the external data in place, the Scheduler activates the SciencePrograms which propose a list of possible targets at the beginning of the night. An aggregated list of field/filter candidates is assembled in SchedulingData. The Scheduler computes the sky brightness conditions for each possible field of the night from the AstronomicalSky model object, integrates this data in the SchedulingData component and the SciencePrograms use this information to generate a list of ranked candidates. Each proposal selects the suitable options after analyzing its current objectives, its own history of granted observations and the weather and sky conditions for each field, and reports the list to the ConductorOptimizer. The conductor takes the aggregated list of ranked targets and calls the ObservatoryKinematicModel for each one of them to obtain the corresponding slew time from the current position and to calculate a slew time rank, combining this value with the rank given by the proposals to compute an overall rank figure for each candidate. The top one is now selected and delivered by the Scheduler to the SimulationKernel, which compares this option against the eventual down time. An observation or down-time is performed, the TelescopeTelemetry simulator is updated with the new coordinates, the time advances, each proposal is notified of the observation in the Scheduler, everything is stored in the database, and the simulator is ready to simulate the next observation.

A graphical description of the behavior for scheduling visits is illustrated in the following SysML activity diagrams.

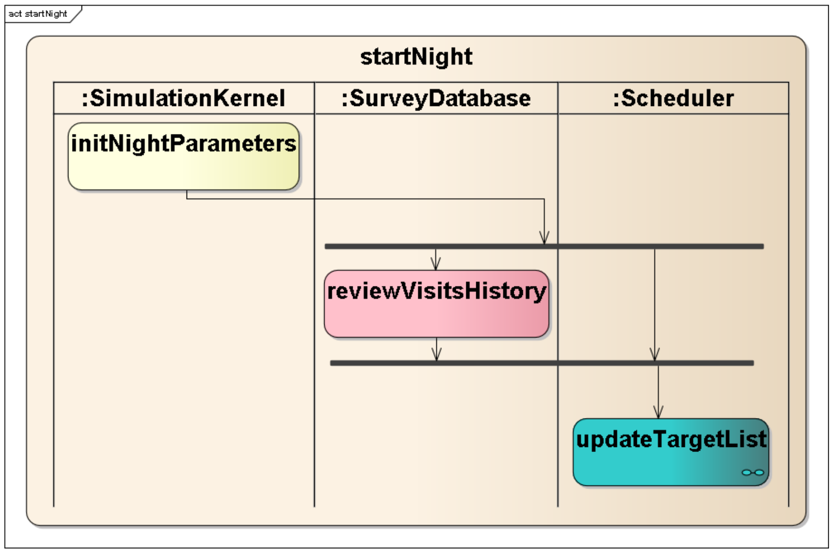


Figure 5. SysML activity diagram for OpSim doing "start night".

Each box represents an activity, and the OpSim component to which that activity is allocated is represented by the column that holds that box. Figure 5 shows the activity "start night".

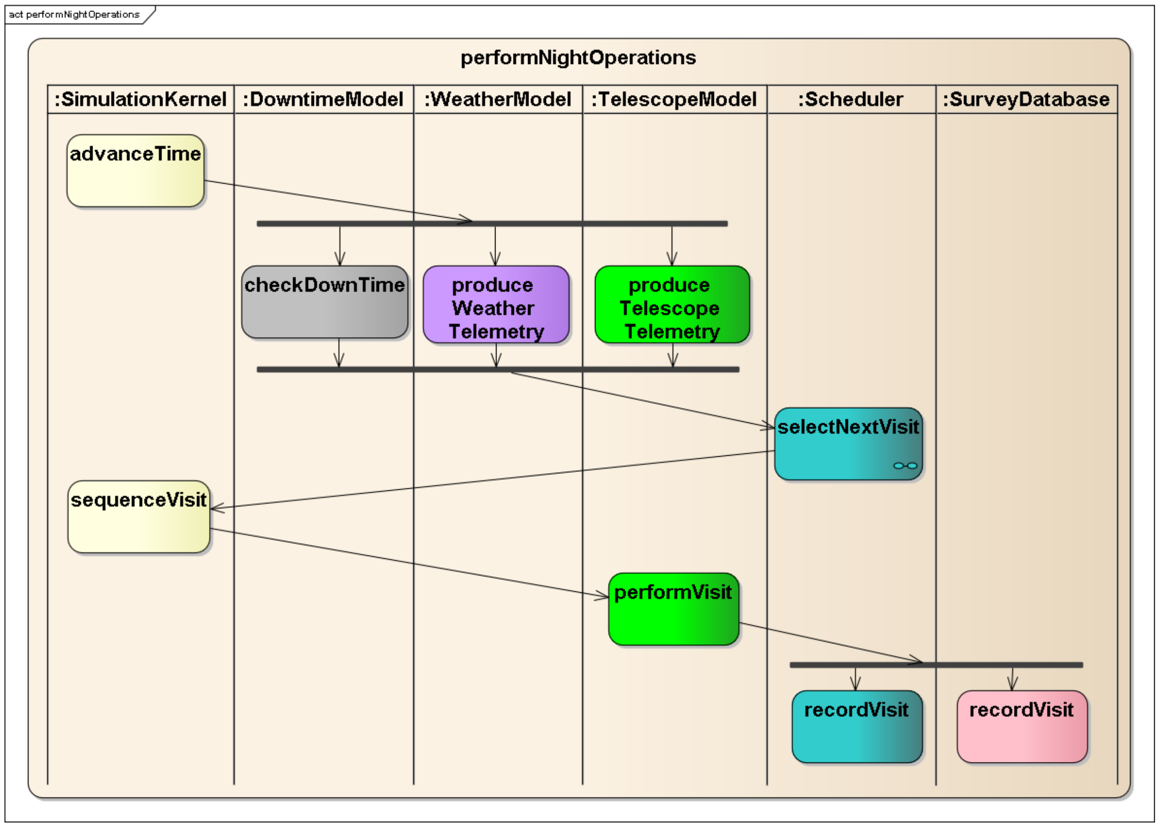


Figure 6. SysML activity diagram for OpSim performing night operations.

Figure 6 shows the OpSim activity of simulating one visit during the night loop. The 3 models simulate the telemetry that define the internal and external conditions, the Scheduler selects the next visit, and the kernel decides to perform that proposed visit or do something else, such as downtime. If the visit is performed, the Scheduler receives that notification for accounting priorities and internal history of observations, and the Survey Database records each simulated action.

5. RANKING THE VALUE OF TARGETS IN THE SCIENCE PROPOSALS

This section is a description of the way that each type of science proposal ranks the targets according to its particular goals and history of observations. In the context of one proposal a higher rank means a higher importance or urgency for a target. In the context of the Scheduler, the rank values must also allow fair comparison of targets from different proposals, so an absolute rank scale for the values needs to be defined. The implemented scale is relative and based on experimentation, but basically a suitable target has a positive rank, where 0.1 is an average value and 1.0 is a very urgent or important target. Higher values are also allowed but should be infrequent.

Coverage Proposals

This type of proposal is designed to get an even distribution of observations on a sky region. The Weak Lensing survey is implemented from this type. The parameters are basically the number of visits per filter that are requested for each field in the region. It gives more ranking to the more needed field-filter combinations. A simplified set of the formulas is shown as an illustration:

$$GlobalNeedFactor = 1 - \frac{GlobalVisits}{GlobalGoal}$$

$$FieldNeedFactor = 1 - \frac{FieldVisits}{FieldGoal}$$

$$FieldFilterNeedFactor = 1 - \frac{FieldFilterVisits}{FieldFilterGoal}$$

$$rank = IdleRank\frac{FieldNeedFactor + FieldFilterNeedFactor}{2GlobalNeedFactor}$$

At the beginning all of the field-filter combinations start with the same *IdleRank* value (0.1 as default), and as the survey progresses the field-filters getting more visits than the average will receive lower rankings and vice versa.

There is a parameter that can enable overflow observations for the completed field-filters, giving them a low but non-zero ranking.

If the duration or the restrictions of the survey are too tight for the goals to be met, the previous equations promote an even progress of the sky region instead of completing fields. There is also the option of promoting the completion of the more advanced fields to get the required depth at the cost of an incomplete sky region. To choose this alternate behavior there are two parameters that control this boost factor to the more advanced fields, so they can receive a higher ranking when they are close to completion.

If look ahead information is available, then the following self-balancing equations can be applied, as an example:

$$targetNeed = \frac{goalVisits - numVisits}{availableTime}$$

$$targetMerit = \frac{targetNeed}{\max(targetNeed)}$$

availableTime is the addition of the future time windows when the target (field-filter) is visible for the science program.

targetMerit gives a normalized range of values

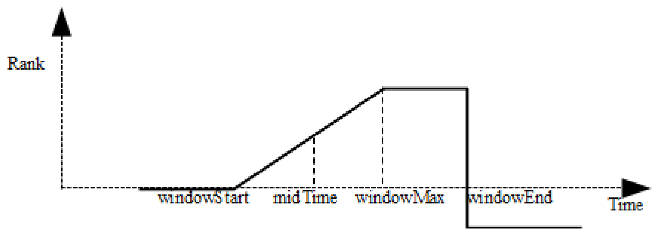
Transient Proposals

This type of proposal is designed to work with transient targets, in which the fields need a sequence of observations at a given set of time intervals. Near Earth Asteroids and Supernova are proposals derived from this class.

In order to configure a transient proposal, a list of N time intervals is needed which will determine the N+1 events for each sequence. Every field in the specified sky region has its own sequence independent from the others. An event in this context is an observation of the field that has to be performed during specific time limits given by the scheduling requirements of the sequence, and with a filter to be determined by the particular algorithm of the proposal.

To achieve the sequence behavior using the same ranking paradigm used in the coverage type proposal, every new sequence starts in an idle state, receiving for its first event a constant rank equal to *IdleRank*. As soon as the Scheduler selects a field from an idle sequence to observe, the proposal changes the state of the sequence into active and it waits for the right time interval to offer the next event. A time-window is used to rank the observation options for the next event to attempt to observe it at the desired moment by the Scheduler.

As the Scheduler is constantly receiving candidates to observe from other proposals, the design specification is that every regular candidate target that is suitable and not urgent has a ranking close to *IdleRank*. Candidate targets with rank values bigger than that are more important or urgent. A target with a rank of 0.0 means it is not required and it is of no use to observe. Following these simple specifications the time-window for the events in a sequence is quite natural.



Three parameters define the three points of the time-window for an event in the sequence. These values are specified as a fraction of the observation interval, so they are adaptive to each particular interval in a sequence. Before *"windowStart"*, observing the event is of no use for the proposal, since not enough time has elapsed since the previous observation in the sequence, therefore the field is given a rank 0.0 and no observations are proposed for this sequence. After *"windowStart"* the field is suitable for observing, so its rank starts to grow with time and the chances of getting selected by the Scheduler increase when competing with other sequences in a less urgent state. *"midTime"* is the ideal moment for the observation, equal to *"last event + event interval"*. The maximum rank is reached at the *"windowMax"*, meaning that it is very urgent to observe because there is the risk of missing the event and this maximum value is sustained until *"windowEnd"*. After that the event is no longer useful so it is given a negative rank that results in the observation being missed. The maximum value is normally 1.0, which is 10 times the idle value, assuring a good chance of observing the fields in the required time intervals when several proposals are in competition.

Depending on the particular requirements for the transient survey, there could be more than one filter to choose in an event. In this case the proposal creates one field-filter candidate for each possible filter in the event for the sequence of this field. There is also a filter rank factor that is added to the base rank from the time-window, and there are parameters to give different importance to the different filters in the case of multiple options.

Another feature implemented in the transient proposals, is the tolerance for some missing events in a sequence. There is a parameter that defines this limit, and as long as a sequence does not miss more than that number of events it keeps progressing; beyond that limit, the sequence is considered lost.

There are some additional parameters to enable the restart of lost sequences, enable the restart of complete sequences, consider the lunation limits, and control the creation of new sequences according to the overall survey progress.

Transient Proposals with Sub-Sequences

This type of proposal is a powerful variant of the transient proposal. Instead of defining only a list of intervals and filter conditions for the sequences, this class of proposal defines several subsequences for each field. There is one sequence per field, but this sequence is actually composed by multiple subsequences running in parallel. A simple case could be the definition of one subsequence per filter, each one with a different time interval. The implementation is however more complex and flexible than that. Each sub-sequence is defined by a number of events with a fixed time interval, a filter, its own time-window parameters and limit for missed events. Each one of these subsequences works in the same way as the sequences in the transient class, but they all run independently proposing field-filter combinations until all of them are complete or any of them is lost, declaring in either case the end of the whole sequence.

Balancing proposal's advance during the survey

To promote the least advanced science proposals against the ones that have achieved higher progress, the following simple self-balancing equations can be applied on the optimizer layer.

$$programProgress = \frac{\sum numVisits}{\sum goalVisits}$$

$$programTime = \frac{elapsedTime}{totalTime}$$

$$programProgressIndex = \frac{programProgress}{\mathrm{programTime}}$$

$$programNeedIndex = \frac{1 - programProgress}{1 - \mathrm{programTime}}$$

$$programBoost = \frac{programNeedIndex}{programProgressIndex}$$

6. COMPUTING THE SLEW COST WITH THE TELESCOPE MODEL

This section gives some details of the telescope model that were implemented in order to compute the slew time in the context of the LSST simulator. Some specific parameter values are presented for clarity, but almost every aspect of this model is configurable.

Telescope Altitude

For this axis a second-order model is considered. There is a lower limit and a higher limit for the telescope altitude position, a constant acceleration, a maximum speed and a constant deceleration. The speed limit, acceleration and deceleration are the same for both directions, and are independent from the azimuth position. The initial and final speeds are considered to be zero, discarding the tracking speed.

If *Distance* is the angular altitude distance and the slew is small enough to not reach the maximum speed, then the peak velocity and time delay are:

$$Vpeak = \sqrt{\frac{2|Distance|}{\frac{1}{accel}+\frac{1}{decel}}}$$

$$delay = \frac{Vpeak}{accel}+\frac{Vpeak}{decel}$$

If the peak velocity reaches the maximum limit, then the delay is:

$$delay = \frac{Vmax}{accel}+\frac{d2}{Vmax}+\frac{Vmax}{decel}$$

$$where\ d2 = |Distance| - \frac{Vmax^2}{2accel} - \frac{Vmax^2}{2decel}$$

Telescope Azimuth

For this axis a similar model is considered, constant acceleration and deceleration with a maximum speed, all the same for both directions. But in this case a cable-wrap is included in the model, allowing a range of movements for more than a full circle. An absolute minimum limit and an absolute maximum limit are defined (-270 and +270 degrees in the current LSST design), and the absolute azimuth moves in this range. As for some target positions there are 2 possibilities, the closest to the current absolute position is chosen for determining the distance and direction for the delay computation.

Rotator Angle

The same constant acceleration model is applied here, and a cable-wrap is used as well. The difference in the rotator model is that the absolute angle limits allow a range of less than a full circle. In the case of the current design for the LSST these limits are -90 and +90 degrees, covering a half circle. When the target angle is outside this range, the current implementation uses the opposite angle instead, so if the target angle was given to put North up, the actual final orientation will have North down. Once the target angle is determined, the kinematic model is applied to calculate the delay time in the same way as the previous axis.

Besides the parameters minimum angle, maximum angle, acceleration, deceleration and maximum speed, this component has an additional one, “follow sky”, which controls the orientation of the image. If this parameter is enabled, the target angle for the rotator will follow the parallactic angle to put North up (or down according to the range possibilities). If this parameter is disabled, the rotator angle is left where it is during the slew, but it tracks during the exposure.

Dome Azimuth

A similar second-order kinematic model is used for the azimuth angle in the dome, with its own maximum speed, constant acceleration and deceleration, with no cable-wrap limits. The model for this component is very simple: the target angle is the same as the telescope, with no observing window or hysteresis, as there is no strategy defined for this in the design yet. Moreover, the simulation results show that the dome delay is not important in the critical path for the overall slew delay. This simplification also implies that the dome is always tracking during the exposures.

Dome Altitude

Dome Altitude is modeled similarly to the telescope altitude, with its own parameters. The delay is associated with the movement of an observing window in the dome. As well as in the dome azimuth model, no window size is considered yet for the same reasons, using for the target angle the same value than in the telescope mount.

Telescope Optics Corrections

For the active optics corrections model, no coordinates for active components are tracked. A very simple model computes the time delay to adjust the optics as a function of the altitude slew in the telescope. There are 2 components in this model: the open-loop corrections and the closed-loop corrections. Both delay components need to be added because the eventual closed-loop correction is always executed after the open-loop correction.

For the open-loop corrections model, the supposition is that correcting from the look-up tables is fast enough or not needed during tracking, so there is a penalty only when correcting for the slew. This penalty is modeled as a constant rate in time/altitude-angle-distance for the whole altitude range.

For the closed-loop corrections, the supposition is that there is a first range of altitude-angle-distance (between 0 and a defined *limit1*) for which no correction is needed. There is a second range (between *limit1* and *limit2*) for which a single exposure is needed for the correction, adding a penalty of a constant delay time. Finally there is a third range (between *limit2* and 90 degrees) for which 2 consecutive corrections are needed producing a longer delay time.

Settle Time

The settle time is a constant delay applied to every slew and accounts for damping the possible vibrations in the mount.

Filter Change

If the target filter is different than the current filter position, a constant delay is applied for this stage.

Readout

This is the delay in the readout for the previous exposure, considered part of the following slew.

Critical Path

As some of the slew activities can be performed in parallel and others in sequence, the total slew delay is the result of the critical path of the individual delays according to the dependencies defined. This dependencies tree is fully configurable. The way to define it is to specify the list of activities that are prerequisite for each one. Our current model defines the following dependencies:

The activities with no prerequisites that can start just after the previous exposure are Readout, Dome Altitude, Dome Azimuth, Telescope Altitude, Telescope Azimuth, Rotator and Filter Change. All these can be performed in parallel.

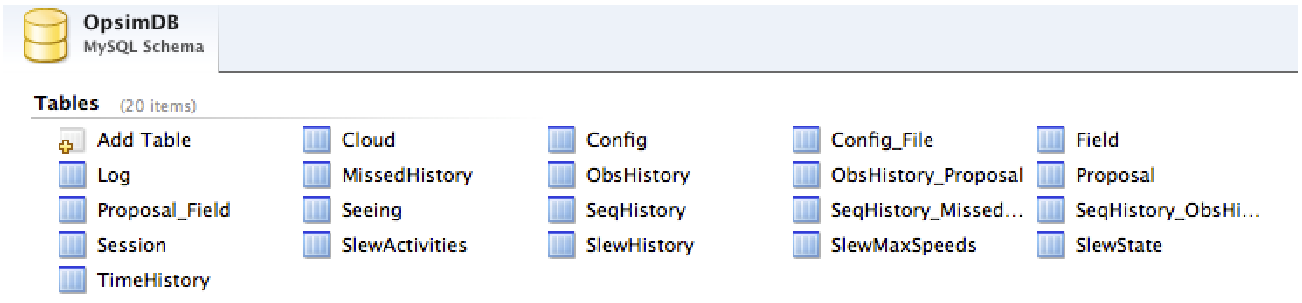
Settle Time and Optics Open-Loop can be executed in parallel and need the mount in position, so each one has as prerequisites Telescope Azimuth and Telescope Altitude.

Optics Closed-Loop needs as prerequisites Optics Open-Loop and the conditions to take images from the sky, which means also Readout, Settle, Filter Change, Rotator, Dome Altitude and Dome Azimuth.

The exposure can be executed after Optics Closed-Loop has finished, meaning that the end of this correction marks the end of the slew.

7. DATABASE

The Simulator architecture section briefly talks about the output of the Operations Simulator. This section details the architecture of the database that holds both the input and output aspects of the OpSim. Additionally there are a number of input parameters like the configuration files relating to proposals that are stored in the database, as well as some pre-calculated seeing and airmass information.



OpsimDB
MySQL Schema

Tables (20 items)

Add Table	Cloud	Config	Config_File	Field
Log	MissedHistory	ObsHistory	ObsHistory_Proposal	Proposal
Proposal_Field	Seeing	SeqHistory	SeqHistory_Missed…	SeqHistory_ObsHi…
Session	SlewActivities	SlewHistory	SlewMaxSpeeds	SlewState
TimeHistory				

Figure 7: OpSimDB v3.x 22 tables

The OpSim v3.x database architecture contains 22 tables. Figure 7 shows a screenshot of the MySQL workbench utility which was used to create the database schema. Following is the description of the input and output tables.

Input Tables of OpSim

- Cloud: keeps track of the Cloud information throughout 10 years of the survey.
- Seeing: keeps track of the Seeing information throughout 10 years of the survey.
- Field: stores the entire visible sky as field centers. The values of fields are pre-calculated and stored in this table during the installation of the OpSim.

Output Tables of OpSim

- Session: the driver table of an OpSim run. A new table entry is created for every OpSim run and is stored in this table. All output tables have a foreign key relationship with this table and output data is identified primarily using the sessionID column of this table.
- Config: keeps track of the various parameters used to drive the OpSim. Configuration file parameters are kept in this table.
- Proposal: keeps track of the various scientific proposals that were used to drive the OpSim.
- Config_File: keeps the raw data of the configuration files used to drive the OpSim.
- Log: keeps the code level log statements. These log entries are used to find errors, warnings and used for debugging purposes.
- MissedHistory: keeps track of the missed observations for an OpSim run and for a field.
- ObsHistory: keeps track of the observations that were taken by the telescope for an OpSim run and field.
- ObsHistory_Proposal: a many-to-many relationship table that keeps track of which observations fulfilled which proposals and vice-versa for an OpSim run.
- Proposal_Field: a many-to-many relationship table that keeps track of which fields were requested for which proposals for an OpSim run.
- SeqHistory: keeps track of the hierarchical information of the various sequences requested for a proposal, for a field for an OpSim run.
- SeqHistory_MissedHistory: a many-to-many relationship table that keeps track of which observations were missed for a sequence and for an OpSim run.
- SeqHistory_ObsHistory: a many-to-many relationship tables that keeps track of observations achieved for a sequence and for an OpSim run.
- TimeHistory: keeps track of the various time events that occur for a night for an OpSim run.
- SlewHistory: a one-to-one relationship table between the SlewHistory table and the ObsHistory table. It keeps track of the Slew associated with each Observation for an OpSim run.
- SlewActivities: keeps track of the various slew activities for a slew.
- SlewMaxSpeeds: a one-to-one relationship table between the SlewHistory table and the SlewMaxSpeeds table. This table keeps track of the various speeds of the instrument for a slew.
- SlewState: keeps track of the initial and final slew states and the various instrument parameters for a slew.

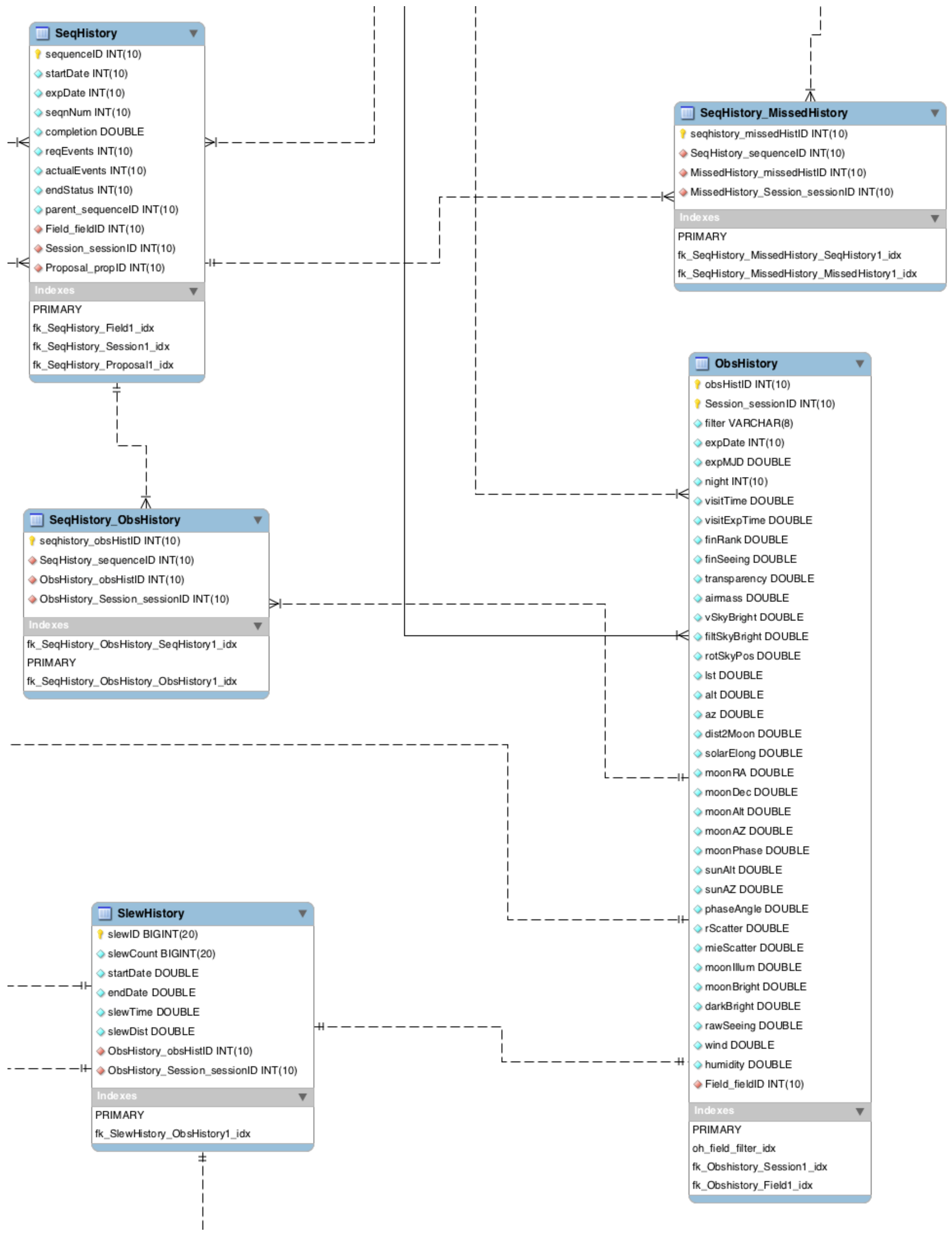


Figure 8: OpSim v3.x database architecture

This architecture makes it possible to query the database in a flexible way thereby assisting the data analysis framework. The use of foreign key relationships reduced the need for duplicating information across multiple tables thereby reducing the size of the database.

8. DATA ANALYSIS

The Observations Database produced from a simulation run, contains all the pointings of the telescope for the 10 years, the conditions (simulated) of each visit, and some technical data that can help engineering analysis. This section focuses on the scientific analysis from a run, and the way to handle the tables in order to validate the simulation, the models, the behavior of the Scheduler, and the levels of success for the particular science cases implemented. For automating this process, a special set of tools have been developed, the Simulation Survey Tools for Analysis and Reporting (SSTAR).

Observing Conditions

A first order diagnostic is to verify that observing condition boundaries are properly observed and that distributions of the values of sky brightness, airmass and seeing for all visits (or broken down by proposal) respect appropriate boundaries. In code validation, one looks for a similar distribution to that of the previous codebase.

Filter Map

The filter map is an inventory of on-sky time and filter used at the time of the visit. The filter used during a visit is plotted at the time of the visit (in hours from midnight local time) as a function of the survey length in days, so each vertical is one day of observing. The enclosing curves are the civil, nautical and astronomical twilight bounds and it is easy to see if the appropriate filters (y & z) are being used when the Sun's altitude is smaller than -12 deg. Dark patches show times of scheduled and random downtime, as well as downtime due to weather. The phase of the Moon is indicated by the white curve along the bottom edge of the plot and is arbitrarily scaled. This diagram is a useful diagnostic of system performance and basic checks on the operation of the telescope – choice of filter with Moon phase (u filter should be used only in dark time), choice of filter in twilight (no visits should occur past 12 deg twilight), and implementation of downtime.

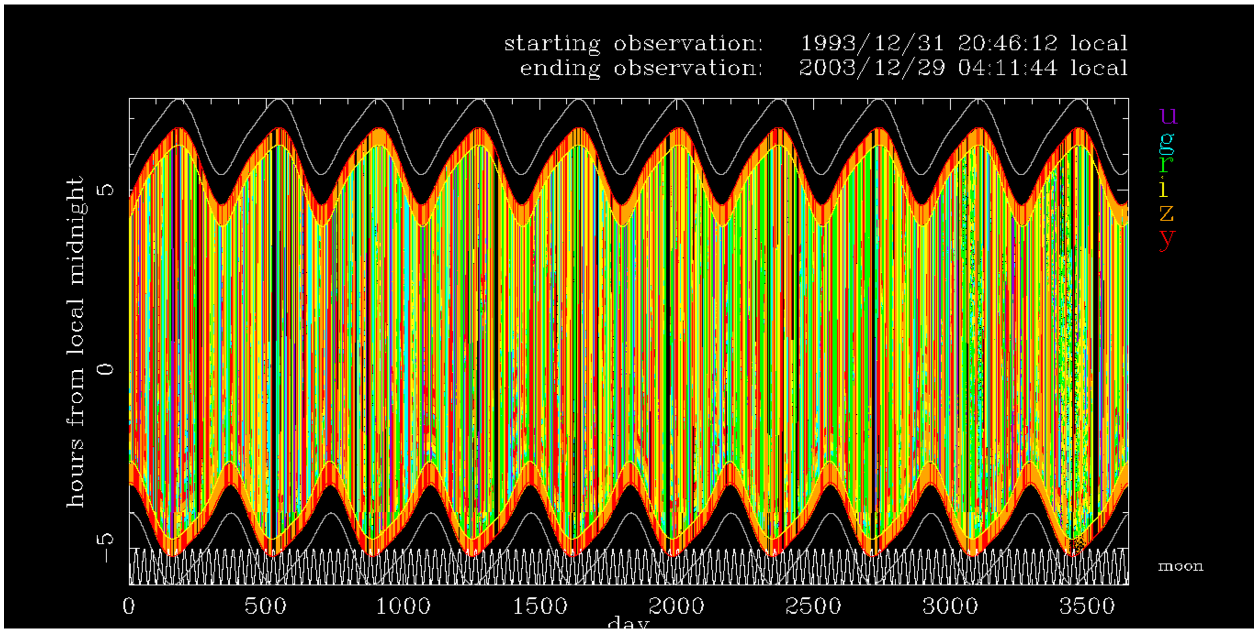


Figure 9. Filter map for opsim3.61.

Sky Brightness

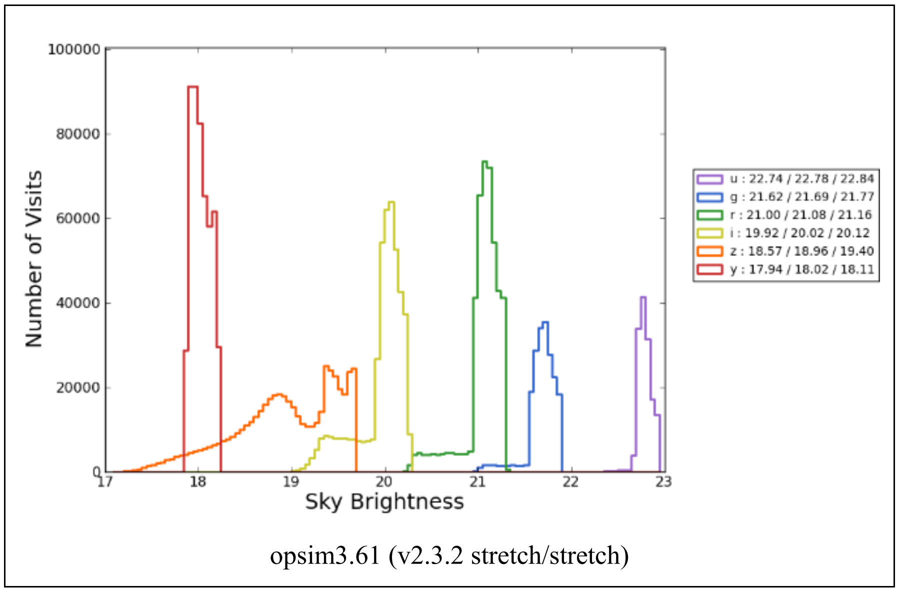


Figure 10. The distribution of visits with sky brightness for each filter. Only visits acquired by observing modes designed to meet the Wide-Fast-Deep program (WFD) number of visits are included. The inset box contains the values of the 25th, 50th (median), and 75th percentiles for each curve.

Seeing

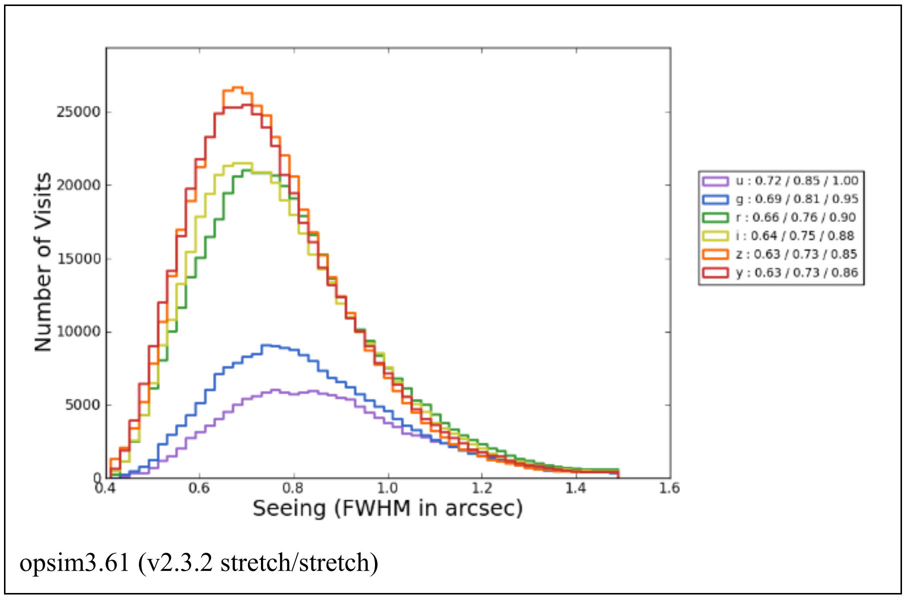


Figure 11. The distribution of visits with seeing for each filter. Only visits acquired by observing modes designed to meet the WFD number of visits are included. The inset box contains the values of the 25th, 50th (median), and 75th percentiles for each curve.

Airmass

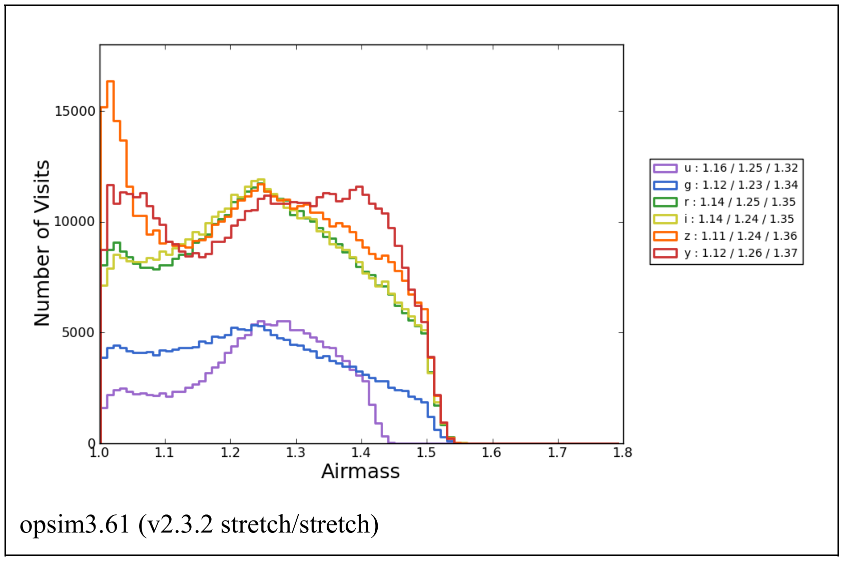


Figure 12. The distribution of visits with airmass for each filter. Only visits acquired by observing modes designed to meet the WFD number of visits are included. The legend contains the values of the 25th, 50th (median), and 75th percentiles for each curve.

Coadded Depth

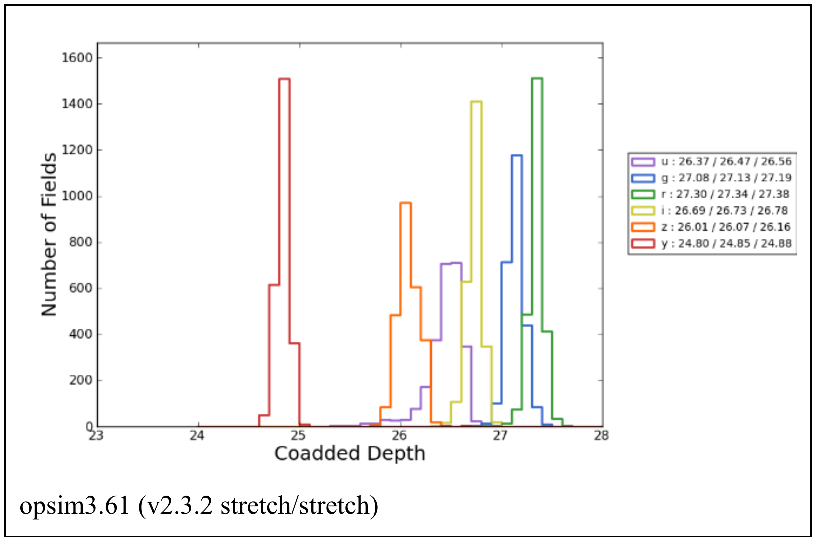


Figure 13. The distribution of fields with coadded depth in each filter. Only visits acquired by observing modes designed to meet the WFD number of visits are included in this plot. The inset box contains the values of the 25th, 50th (median), and 75th percentiles for each curve.

Joint Completeness

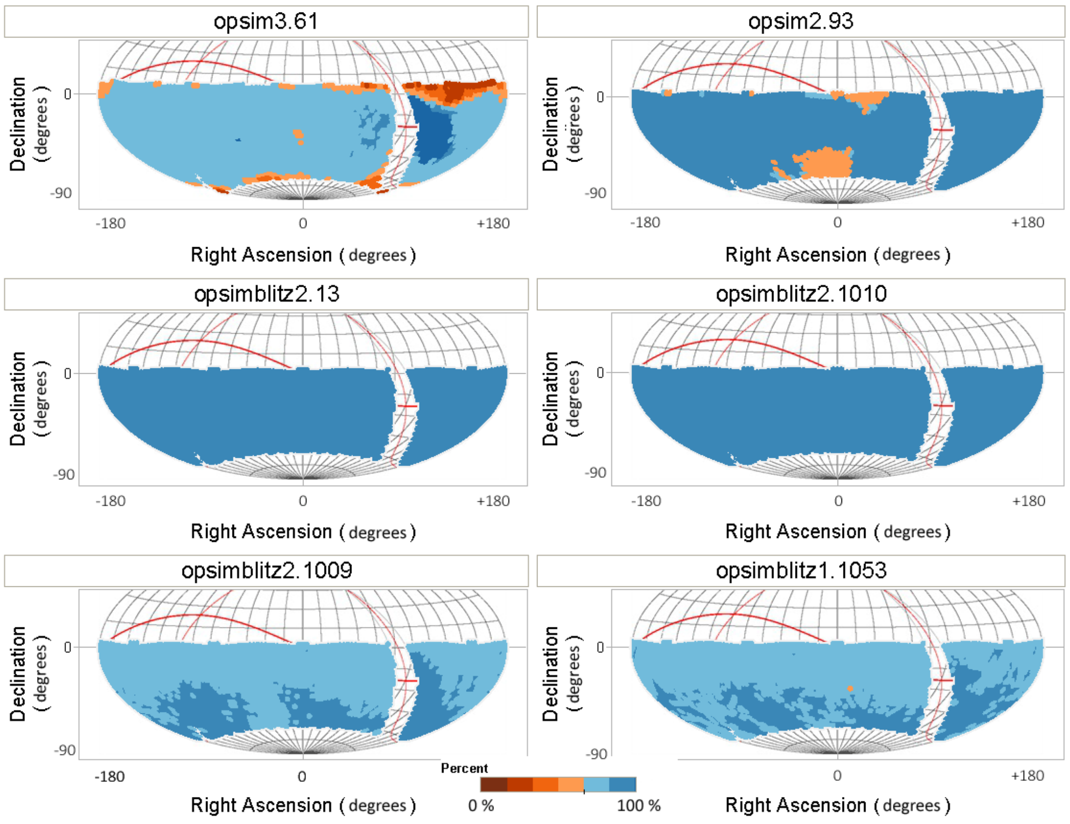


Figure 14. Sky maps of circular fields in Hammer-Aitoff projection. The color indicates for a particular field the value of joint completeness.

9. CONCLUSIONS

The problem of scheduling the LSST, with its one-of-a-kind set of constraints has been addressed with the Operations Simulator. The capabilities of the tool have grown with time, as it approaches mimicking and accommodating real observing conditions. Particularly notable is the detailed modeling of the telescope performance, with parameters that can be tuned to change with real situations as they present themselves. By design, the tool is adaptive, and makes decisions based on the state of the observatory, environmental conditions, observing priorities (with attention to timing requirements), and the history of past observations.

The Operations Simulator has proved to be a powerful tool since the early stage of the LSST project. It has played a key role in several early design decisions for the observatory, including site-selection and field-of-view requirement. In addition to being utilized for experimenting with the design of the survey, this tool has been utilized for systems engineering studies, where many simulations have helped to analyze the impact on the science and costs for different design configurations.

The Scheduler prototype developed inside OpSim is fundamental for reducing the risk on the design of the OCS Scheduler, which is a critical component of the Control System. An evolved version of OpSim will be the perfect environment for testing and evaluating the OCS Scheduler during its development cycle. Operationally, LSST will need both a Scheduler that drives the observing decisions, as well as a Simulator *using identical logic* for longer term strategic planning, situation analysis and performance changes in the observatory, and for exploring mid-course changes in the survey driven by changes in the scientific horizon. Further development will be geared towards meeting these dual needs, and also towards exploring improvements that can produce a survey design that is scientifically most efficacious.

REFERENCES

[1] Francisco Delgado, et al. “LSST Operation Simulator Implementation”. Proc SPIE vol 6270 (2006).
[2] Francisco Delgado, et al. "The LSST OCS scheduler design". This proceedings.
[3] Nicholas Kaiser, et al. “Pan-STARRS: A Large Synoptic Survey Telescope Array,”. Proc. SPIE vol 4836 (2002).
[4] LSST Science Requirements Document (http://www.lsst.org/Science/docs.shtml).