LSST Scheduler
Operations Simulator

Francisco Delgado
CTIO/LSST
Operations Simulator

- Operations Simulator
- Verify the specifications of LSST hardware and survey against SRD
- Design the set of science programs
- Experiment scheduling algorithms and strategies
- Systems engineering trade off studies
- Prototype for OCS Scheduler
Main Specific Goals for OpSim 3.0

• Keep capacity of running OpSim 2.x baseline survey
• Implement look-ahead capabilities
• Increase flexibility for algorithm experimentation
• Expand capacity for scripted cadences

• Evolve OpSim-scheduler into OCS-scheduler
• Improve Observations Database structure and performance
• Improve architecture and performance of the code
• Achieve deterministic behavior
OpSim v3 Look Ahead Window

SchedulingData Component

- Fixed size time window: current technical tests for 30 and 60 days
- Fixed granularity: current technical tests for 5 and 10 minutes resolution
- The look-ahead window manager maintain the size of the look-ahead window, pre-computing additional nights and releasing past nights as the simulated time advances.

- The pre-computed data is used by the science programs to obtain the possible targets and to rank the merit of them.
- Once the target is selected for a visit, the data is recalculated for better precision.
OpSim v3 Look Ahead Data

SchedulingData Component

- A software component that stores look-ahead information for the aggregated list of relevant fields for a given cycle.

- SchedulingData.alt[field][t]
- SchedulingData.az[field][t]
- SchedulingData.pa[field][t]
- SchedulingData.airmass[field][t]
- SchedulingData.skybrightness[field][t]
- SchedulingData.visible[program][field][filter][t]
- SchedulingData.visibleTime[program][field][filter]
Scheduling Visits

Dynamic and adaptive process for each Visit:

– Each science proposal analyzes its assigned sky region, and selects the candidate targets that comply with its requirements for airmass, sky-brightness and seeing.

– Each proposal computes the scientific merit for each target according to its own distribution and cadence requirements.

– The observation scheduler combines all the targets and invokes the telescope model to compute slew cost for each one.

– The scheduler computes the overall rank and select the best.

\[
\text{Rank} = \text{CoaddedMerit} + \text{SlewBonus} \times \frac{a}{(SlewTime+b)}
\]

The new architecture allows room for different algorithms.
Area Distribution Programs

- Designed to obtain uniform distribution
- Basic parameter: goal visits per filter

\[
ProposalNeedFactor = 1 - \frac{ProposalVisits}{ProposalGoal}
\]

\[
FieldNeedFactor = 1 - \frac{FieldVisits}{FieldGoal}
\]

\[
FieldFilterNeedFactor = 1 - \frac{FieldFilterVisits}{FieldFilterGoal}
\]

\[
FieldFilterMerit = IdleRank \times \frac{FieldNeedFactor + FieldFilterNeedFactor}{2 \times ProposalNeedFactor}
\]

- Field-filters receiving visits reduce their rank, while not observed Field-filters increase their rank.
Area Distribution Proposals with Look Ahead

\[
targetNeed = \frac{goalVisits - numVisits}{availableTime}
\]

\[
targetMerit = \frac{targetNeed}{\text{max}(targetNeed)}
\]

– 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
Science Programs balance

\[ \text{programProgress} = \frac{\sum \text{numVisits}}{\sum \text{goalVisits}} \]

\[ \text{programTime} = \frac{\text{elapsedTime}}{\text{totalTime}} \]

\[ \text{programProgressIndex} = \frac{\text{programProgress}}{\text{programTime}} \]

\[ \text{programNeedIndex} = \frac{1 - \text{programProgress}}{1 - \text{programTime}} \]

\[ \text{programBoost} = \frac{\text{programNeedIndex}}{\text{programProgressIndex}} \]
Sequence Possibilities

- One single sequence per field
- Multiple subsequences per field, different filters
- Option for collecting pairs of visits in any subsequence
- Option for combining area with time distribution
- Option for collecting deep drilling sequences, back-to-back visits changing filters
- Option for nested subsequences
Time Distribution Programs

- Designed to obtain specified intervals
- Basic parameter: time window for visits interval

Each field has a sequence of visits with time intervals.
- The rank is calculated applying the time window to the time elapsed since the last visit to that field.
OpSim v3

Sequences look ahead filtering

– A science program with sequences evaluates the look ahead visibility of the field-filter series of visits given a start time.
– A list of possible start times is populated for each sequence.
– The goal is to start only feasible sequences
Next Efforts on OpSim

Look ahead capabilities (efficiency)
Algorithm experimentation
Dithering algorithms
Improve sky brightness and clouds model
Improve the usability and efficiency in the DB
Summary

Powerful tool for survey designing and systems engineering

OpSim was key on site-selection, telescope-camera specifications validation, and finding a survey that fulfilled the science requirements.

OpSim-Scheduler as a prototype for OCS-Scheduler, reducing the risk on a critical component.

OpSim can be evolved into an operational tool for survey assessment and planning.
End of Presentation
OCS Table of Contents

- Observatory Control System context
- MBSE methodology
- OCS Structure
- OCS Interfaces
- OCS Scheduler
- OCS Validation
- Operations Simulator supporting OCS Design
OCS Context

Observatory Control System (OCS) Software
Master Control and System Coordination Survey (automatic and dynamic)
Monitoring Maintenance

Middleware and EFD covered in other talks
The OCS design is developed following a MBSE methodology.

The language is SysML.
The tool is Enterprise Architect.

The model captures and relates:
• Requirements
• Interfaces
• Overall System Architecture
• Components Structure
• System Behavior
• Operational Definitions

Document 9336 “Using SysML for MBSE Analysis of the LSST System”
MBSE methodology
Triad composition between requirements, structure and behavior
Structure is composed of the following main components:
1. OCS Software Application
2. OCS Software Operator
3. OCS Software Monitor
4. OCS Software Telemetry
5. OCS Software Maintenance
6. OCS Software Scheduler
7. OCS Software Sequencer
8. OCS Software Remote

Scheduler identified as most risky component of OCS because it is first large astronomy project that uses this concept to that scale
Each OCS requirement is traced to at least one OCS structure block (no orphan requirement).

Each OCS structure block is traced to at least one OCS requirement.
OCS Software Structure Internal Interaction between Components

Internal Flow and Ports determined in SysML
OCS to Camera Interface Model Example
This sequence shows exchange of commands (black), telemetry (red) and events (orange) between the OCS and camera to take an image.
OCS controlling the observatory example with commands and events

OpSim Scheduler 2013 July • Tucson, Arizona
These concepts come from the Opsim software development:

- Fully automatic operation
- Dynamic adaptation to environment and observatory conditions
- Sky field map, tiling regions
- A target is a sky field/filter combination
- Weather and sky conditions from telemetry
- Sky brightness dynamically modeled for each sky field
- Comprehensive observatory kinematic model for slew time optimizations
- Fully configurable set of concurrent competing science programs
- Some science programs optimize visits depth distribution
- Some science programs optimize visits interval distribution
- Some science programs push for time interval/filter sequences
- Scripted visit sequences are also possible on a science program
- Next targets selected according to a ranking schema
- Target rank balances science value and time cost
- Observations database
- Scheduling parameters can be fine tuned to direct the survey
OCS Software Scheduler Traceability to OCS Requirements

from OCS reqs

from OCS structure
OCS Software Scheduler Internal Interaction between Components
The following slides show an example of the triad validation methodology for the OCS design.

From LSST Observatory all the way to the OCS Scheduler Kinematic model structure component, flowing top down through the corresponding requirements and behavior.
Perform Survey Activity

Goal: Provide summary background on telescope requirements and performance analyses used for Hexapods and Rotator Requirements
Perform Science Observations Validation

Data Collection partial break-down.

The sub-activity “Select Target” is allocated to OCS. The associated functional requirement is “Schedule Survey”.

OpSim Scheduler 2013 July • Tucson, Arizona
Select Target Validation

The requirement “Optimize observing time” is contained in the previous “Schedule Survey”. OCS partial break-down.

OpSim Scheduler 2013 July • Tucson, Arizona
Rank Targets validation

Scheduler break-down.

“Estimate slew delay for candidate observations” is derived from the previous “Optimize observing time”.
• Each OCS structure block traced to at least one OCS requirement
• Each OCS requirement also traced to at least one OCS structure block (no orphan requirement)
• Behavior diagrams validating key requirements with structure
• Interfaces identified
• Following the Design validation methodology:
  – Discovery of missing and unsatisfied requirements
  – Validation of design choices
  – Automatic document generation
  – Consistency
Opsim Database v3.0

Srinivasan Chandrasekharan
NOAO/LSST
July 18th 2013
Why we need the DB

- Different Simulations
  - Session
- Input to the Simulations
  - Cloud
  - Config
  - Seeing
  - Proposal etc.
- Output of the simulations
  - ObsHistory
  - Slew Information
  - Proposal & ObsHistory
  - Proposal & Field etc.
- Ease of access for post processing, less joins
- Make DB access faster
What was lacking in the old schema?

- No relationships between tables
- MyISAM tables cause table locks
- Performance degradation due to MyISAM
- ObsHistory, Proposal, Field relationship important for post processing
  - Harder in the older schema
Implementation

- MySQL workbench
- Data access library is a hybrid MySQLdb and SQL alchemy for ORM
Demo