Scheduling

Purpose

- Determine best choice of Group to execute under given sky conditions with reference to priority and observability.

- Scheduler on LT is a passive system
  - It generates schedule only on request by RCS.
  - RCS then executes or may do something else.

- Alternative is an active scheduler.
  - Generates a schedule or plan and gets rest of system to implement it.
  - This is what most people seem to assume happens.

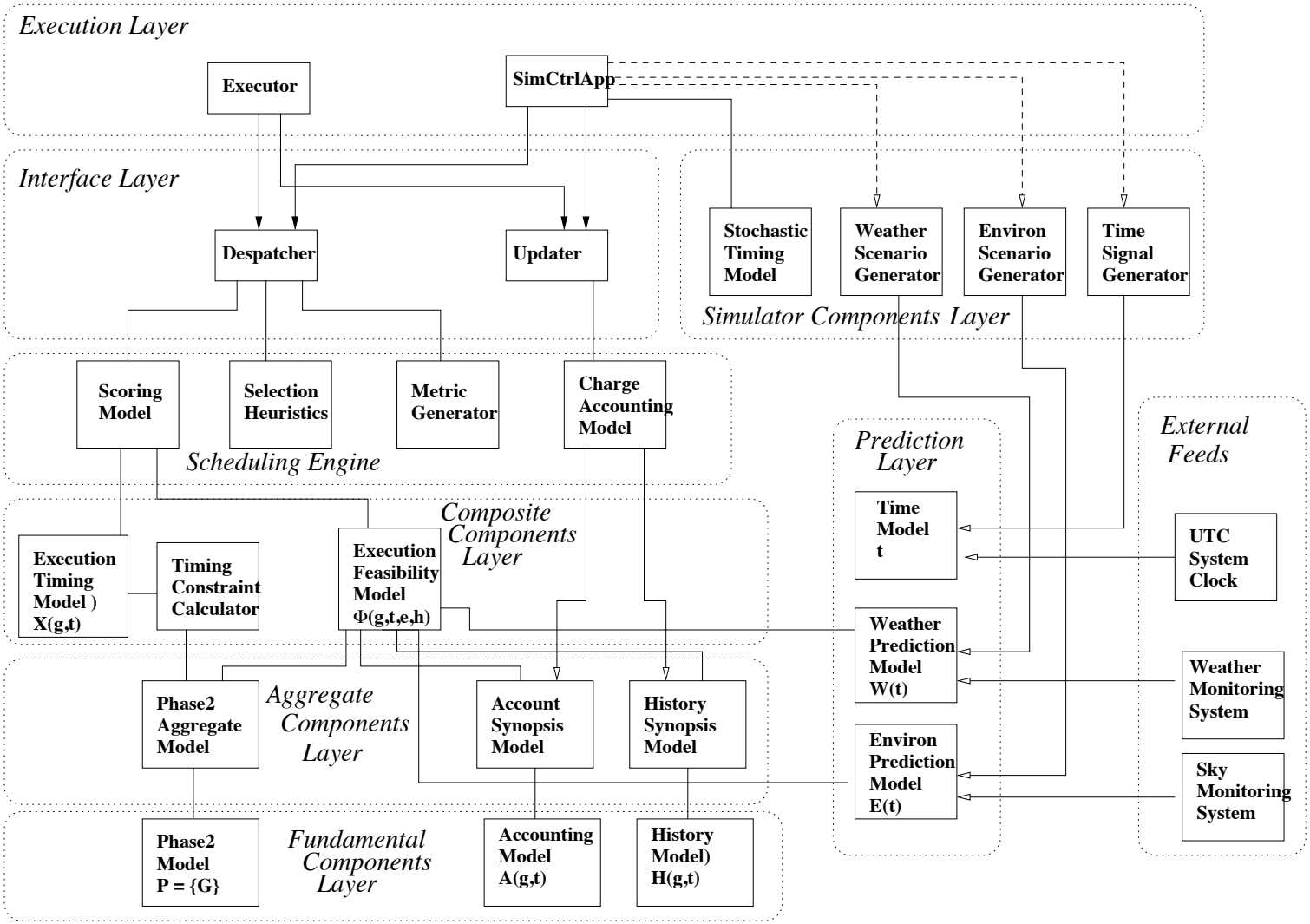
TYPES OF SCHEDULER

- Despatcher
  - Selects an individual group on request which is suited to conditions, feasible and in some way optimal.
  - Always picks the best group *at that time*.
  - The executed sequence however may not be optimal.
- Look-ahead
  - Generates a sequence of groups which are then passed back to requestor as needed
  - Optimal sequence at the time – may not be same as sequence generated by despatcher.
  - Sequence may break due to environmental changes and become sub-optimal.

Background

- Adaptive Optimal Telescope Scheduling
  - http://1drv.ms/1xx9tBd
  - Chapters 1 (intro) and 5 (architecture)
- The LT scheduling system SMS and the scheduler (BDS) is in the *ngat.sms* package.
- Older schedulers and OSS classes are in the defunct
  - ltdevsrv:~*dev/src/oss/*
- Current despatch scheduler:-
  - *ngat.sms.bds.BasicDespatchScheduler*
- Various Look-ahead schedulers also implemented
  - TLAS
    - ngat.sms.tlas.TestLookAheadScheduler
  - QLAS
    - ngat.oss.simulation.QuantumLookAheadScheduler

ARCHITECTURE



SYNOPTIC MODELS

- Generally referred to as *the cache*.
- Phase2CompositeModel
  - Active and unexpired groups and details of:-
    - Proposal, PI User, TAG, Sequence
- AccountSynopsisModel
  - Details of accounts per group
  - Just balances, not transactions
- HistorySynopsisModel
  - Details of execution history per group
  - Last successful exec and count
- Individual models are collected together and served by a SynopticModelProvider.
- Models are connected to base models.
  - Phase2Model, AccountingModel, HistoryModel
  - They receive callbacks when the base models change.

Additional Synoptic Models

- InstrumentSynopsisModel
  - InstrumentSynopsis for each instrument
  - Connected to InstrumentRegistry to obtain InstrumentStatus and capcabilities.
- TelescopeSystemsSynopsis
  - Details of telescope state and capcabilities
  - Connected to TelescopeStatusProvider and TelescopeCapabilitesProvider

Computational Models

- ExecutionResourceUsageEstimator
  - Calculates execution time for a group
    - By reference to its sequence components
    - Fairly crude – needs improving using gathered statistics – (see: rcs task log)
    - *ngat.sms.bds.StandardExecResourceUsageEstimator*

- ExecutionFeasibilityModel
  - Determine if a group can execute:-
    - At a specified time
    - Under specified sky conditions
    - Given its execution history
    - Given its (Proposal's) account balance
    - *ngat.sms.models.standard.StandardExecutionFeasibilityModel.*

More Models

- ChargeAccountingModel
  - Calculates how much time a group should use
    - Based on figures published on website (overheads)
    - Needs keeping upto date as things change
- Scoring model
  - Built in to despatcher but has in the past been extracted as a separate model – would be better.
  - Used to create group scores.
- Selection heuristics
  - Used to decide which of the scored groups to select.
  - Currently embedded in despatcher.
  - In the past several models have been tried.
  - Currently we select highest ranked from scoring model.

Operation (Sweep)

- *[Telemetry (sweep starting)]*
- Find any fixed groups
- Determine sky conditions
- For each group {
  - Get execution history synopsis
  - Get Account synopsis
  - Check feasibility
  - Either add to candidates or reject with reason
- }
- Test each candidate in lists (primary, background, fixed) {
  - *[Telemetry (candidate group)]*
  - Calculate score
  - Find highest score
- }
- Highest scoring group is selected
- Obtain history ID reference (from history model)
- *[Telemetry (selected group)]*
- Return selected group

Feasibility Criteria

- StandardExecutionFeasibilityModel
  - Timing constraints
    - Inside window, start and end times, repeat count
  - Observing constraints
    - SkyBrightness at target(s) location
    - Airmass
    - Hour Angle
    - Seeing
    - Photometricity
  - Implicit constraints
    - Accounts – sufficient time
    - Enablement (group, proposal)
    - Target visible
    - Time to sunrise
    - Instrument online, functioning, enabled
    - Rotator settings feasible
    - Acquisition instrument available

Statistics

- Last night (3-10-2014)
  - Number of groups in cache: 1142
  - Max candidates per sweep: 160
  - Time per sweep: 7-9 sec
- Number of active groups rises over time
- Sweep time also increases
- Add graph here….