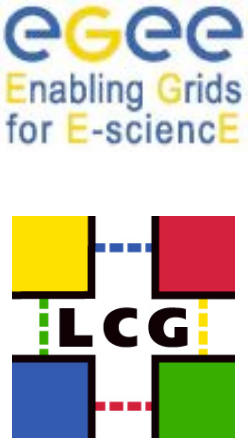
SAM Submission Framework



Piotr Nyczyk

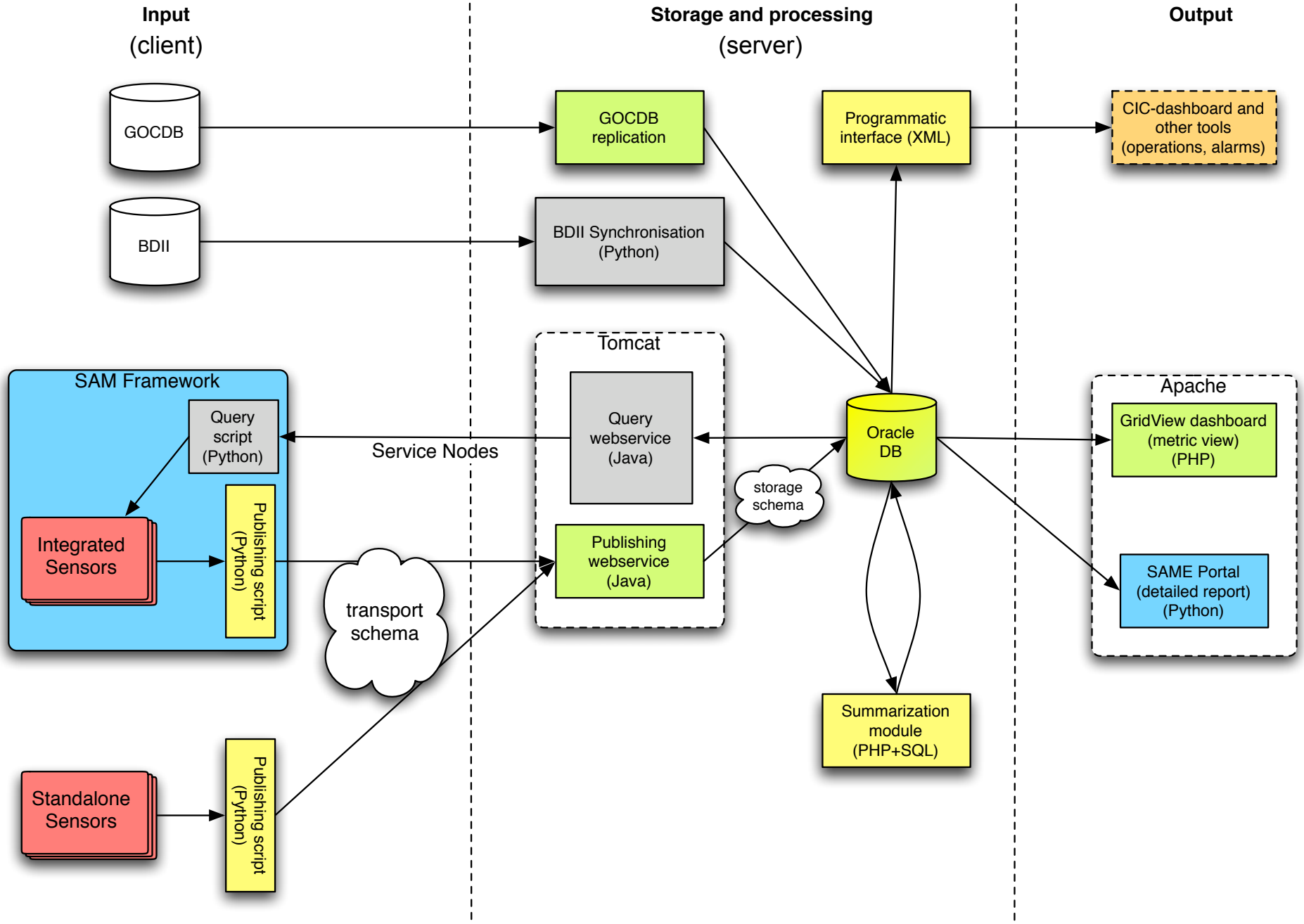
SAM Review

CERN, 2007

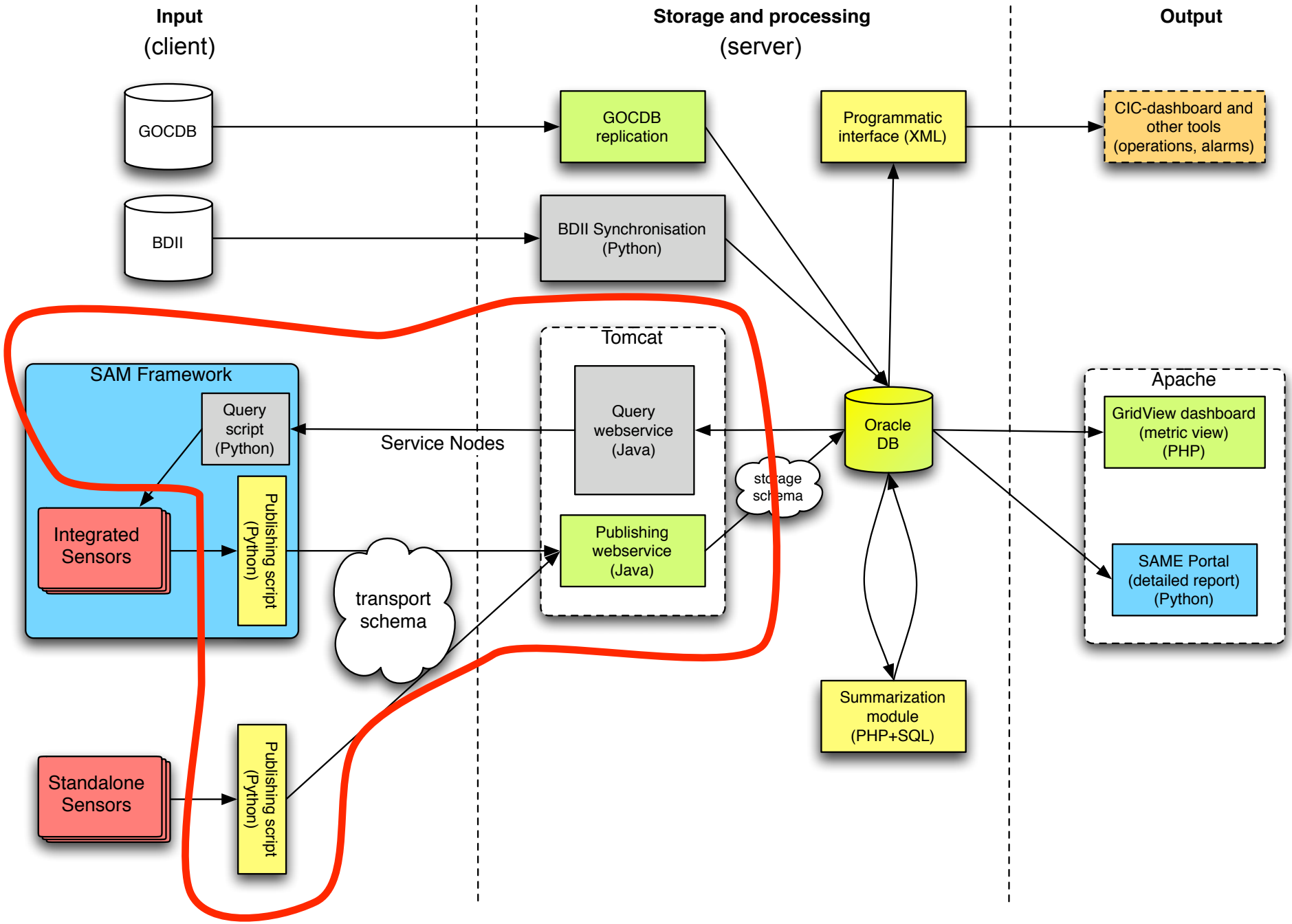
Outline

- SAM Overview
- Introduction
- Terminology
- Architecture
- Status of interfaces and services
- Open issues

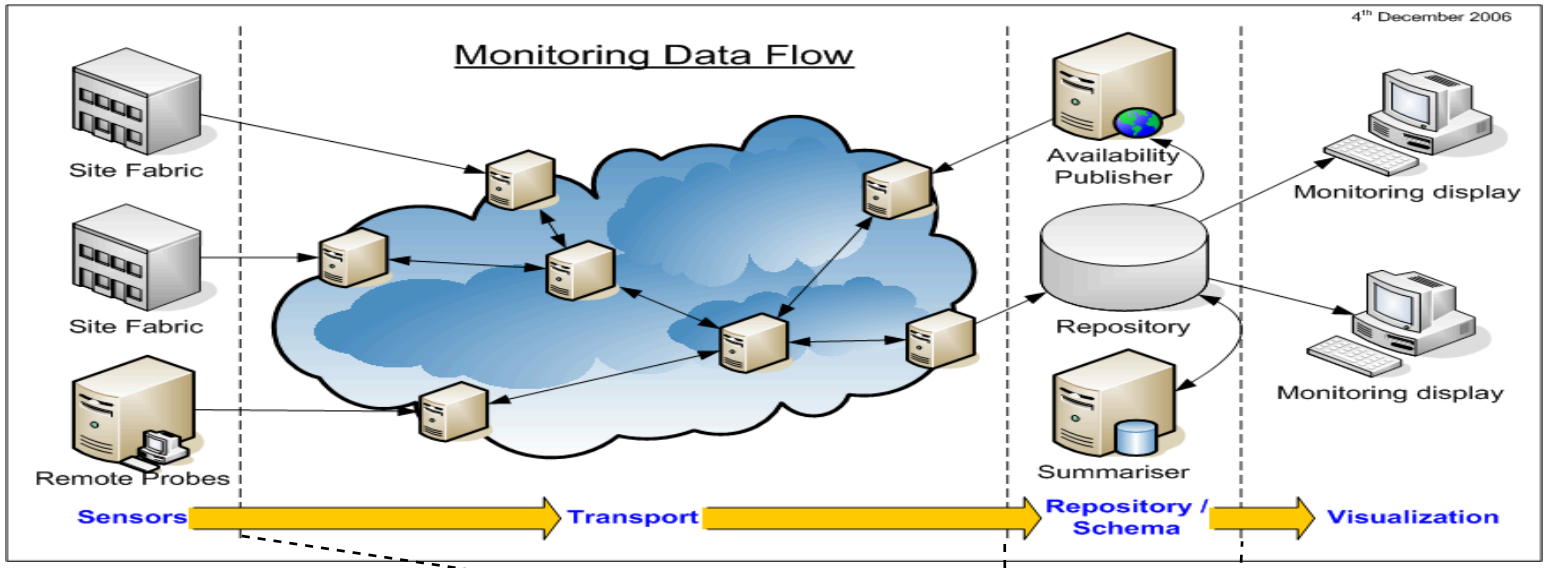
SAM Overview

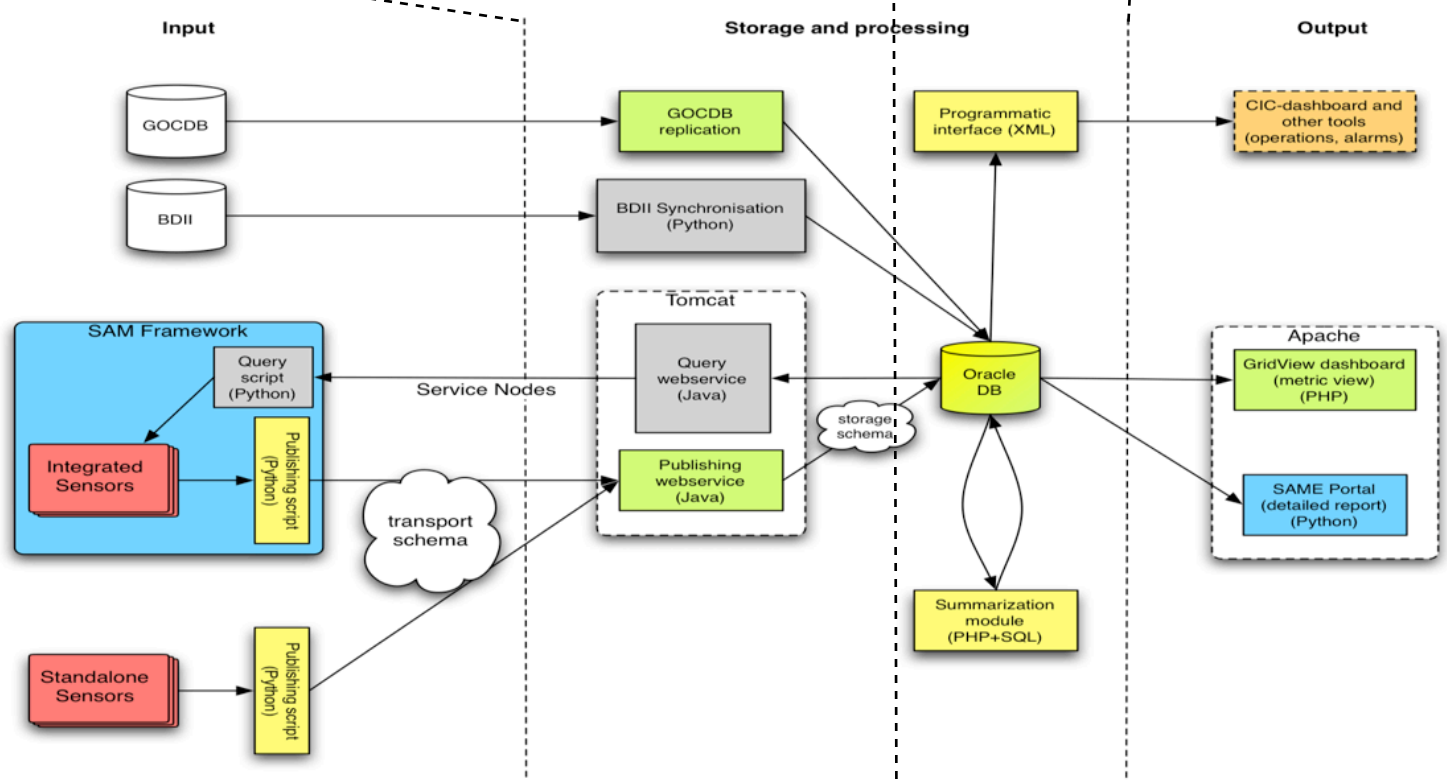


SAM Overview



SAM vs. Monitoring WG





Introduction

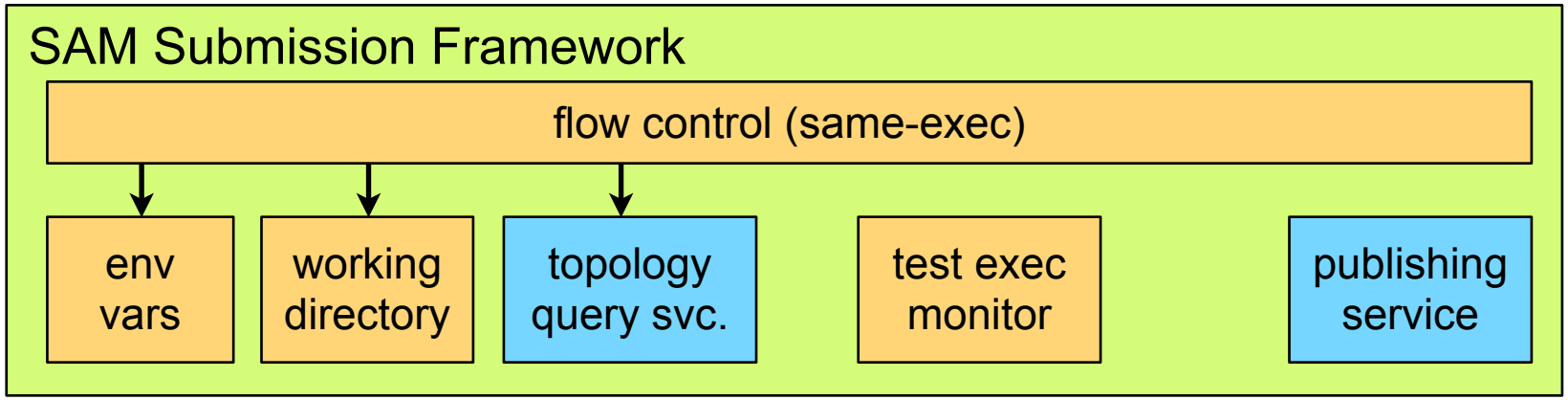
- SAM Framework provides
  - command line interface to run sensors
  - common environment and sensor state storage (working directory)
  - high-level sensor execution workflow control (timeouting, parallelism)
  - unified access to “Topology Database” (SAM/ GridView DB based on GOCDB and BDII)
  - unified publisher interface
- SAM Framework DOES NOT provide
  - grid credentials management (!)
  - inter-services dependencies checking
  - inter-tests dependencies

Terminology

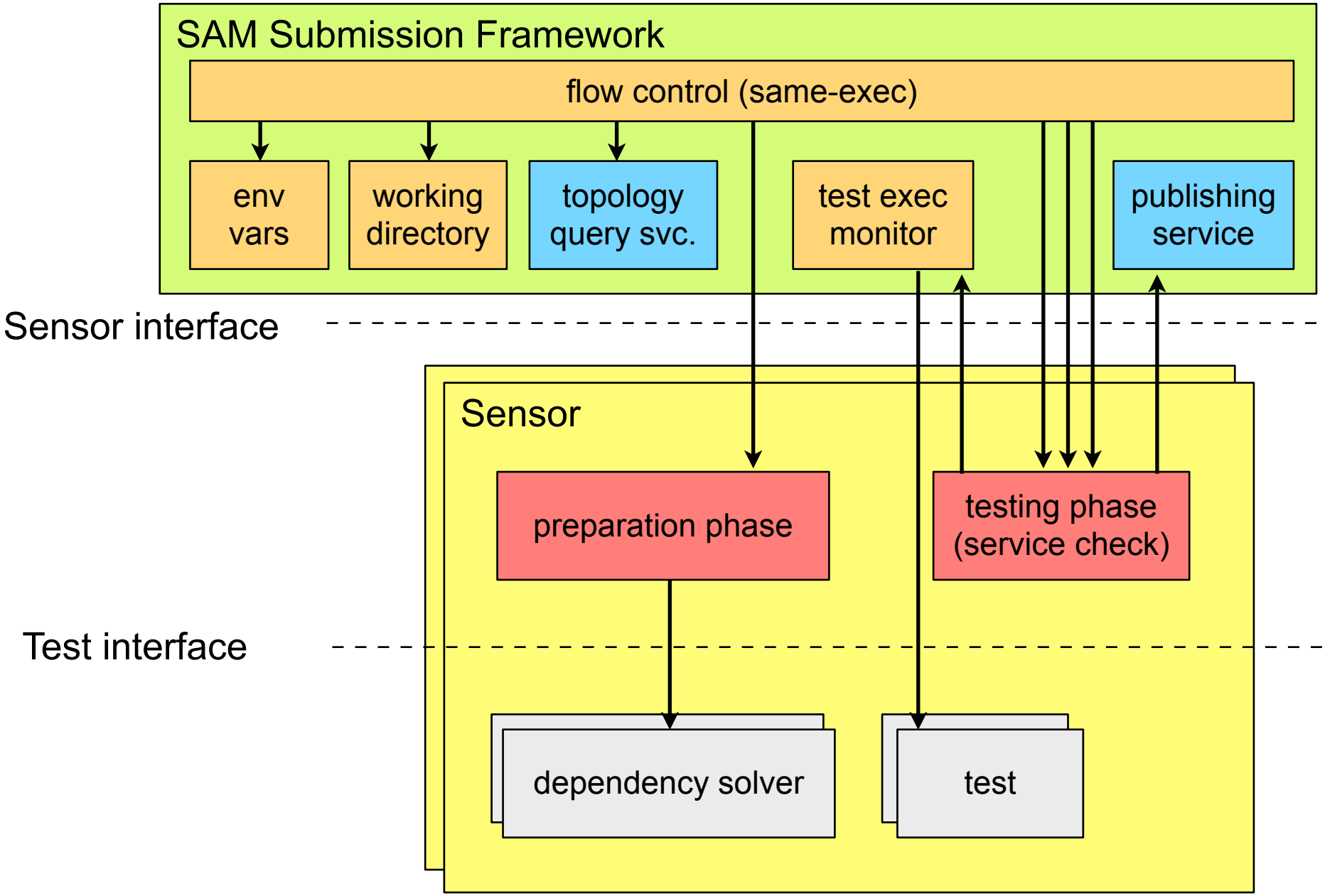
- Sensor - collection of tests usually grouped by related functionality like service type (MWG: *probe)*
- Test - smallest unit returning a single measurement recognised by SAM(MWG: *metric)*
- Topology query - web service interface to SAM DB for information about sites, service instances, VOs...
- Sensor state - files kept in sensor's working directory across calls to SAM Framework, examples: job IDs, intermittent test results, log files, cached information

Architecture

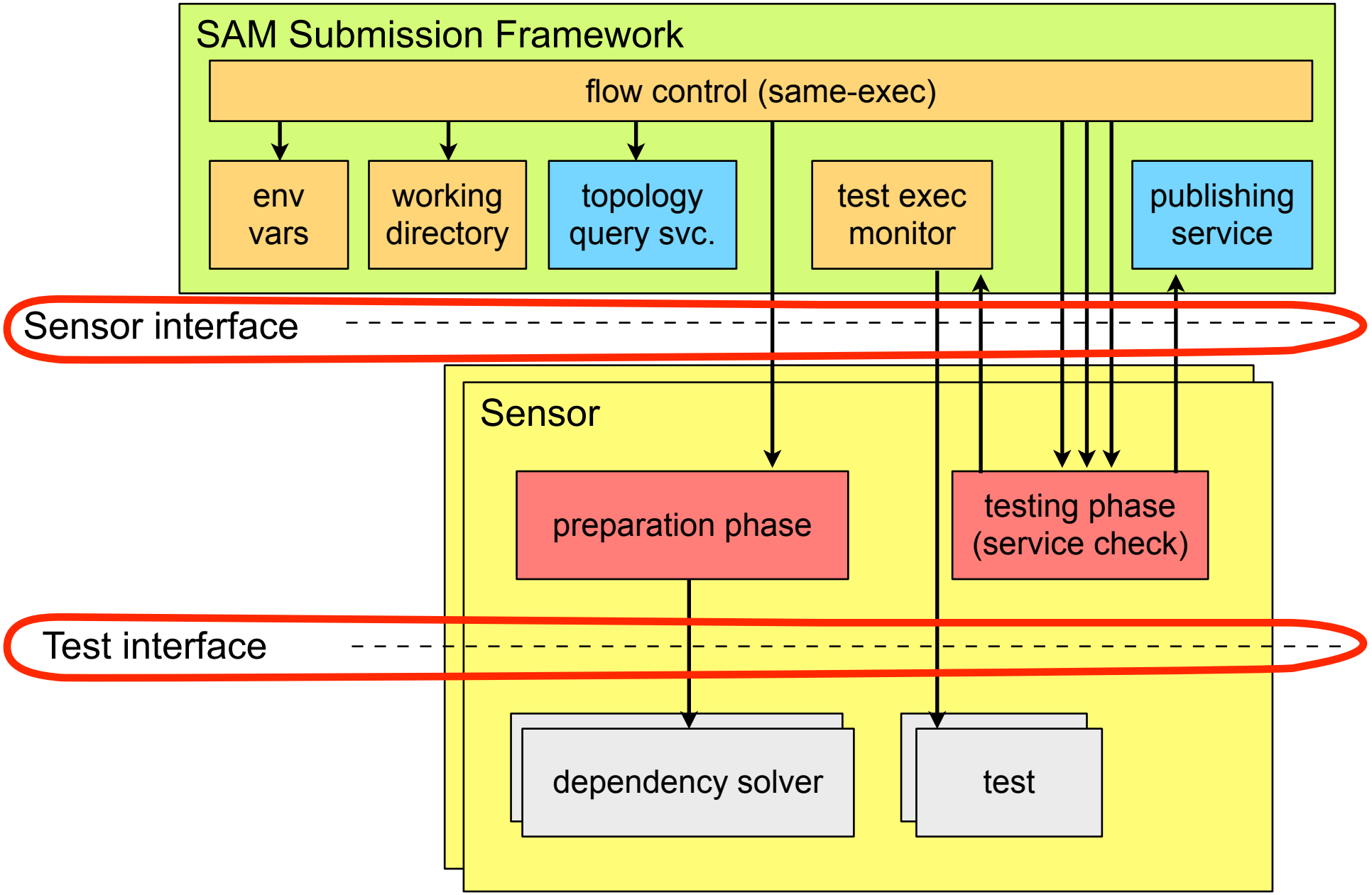
Architecture



Architecture



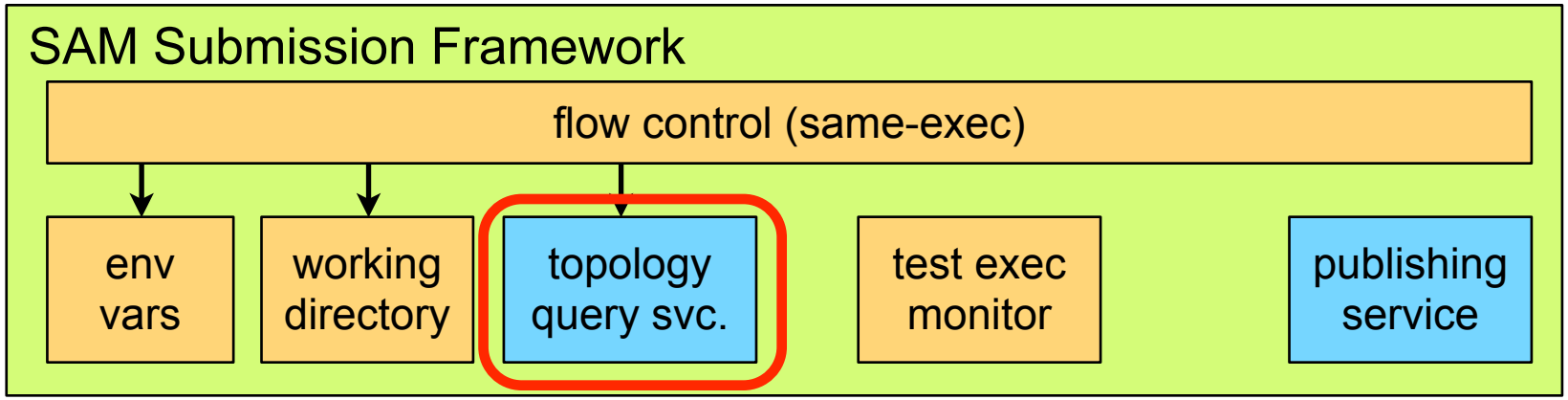
Interfaces


SAM Submission Framework
flow control (same-exec)
env vars
working directory
topology query svc.
test exec monitor
publishing service
Sensor interface
Sensor
preparation phase
testing phase (service check)
Test interface
dependency solver
test

Interfaces

- Sensor interface:
  - defines: sensor's directory structure, topology query protocol, publishing protocol, command line interface (two-way), execution workflows (operations)
  - almost unchanged from the beginning of SAM
  - proved to allow easy separation of Sensors from Framework (even for external developers)
  - defines 4 basic operations (workflows) now: *submit, publish, status, cancel* (+ few special internal operations)
  - publishing protocol used as basis for Monitoring WG's common probe format
- Test interface
  - very simple: STDOUT/STDERR (HTML dump), exit code
  - almost unchanged from times of SFT (except naming convention for tests and status variables)

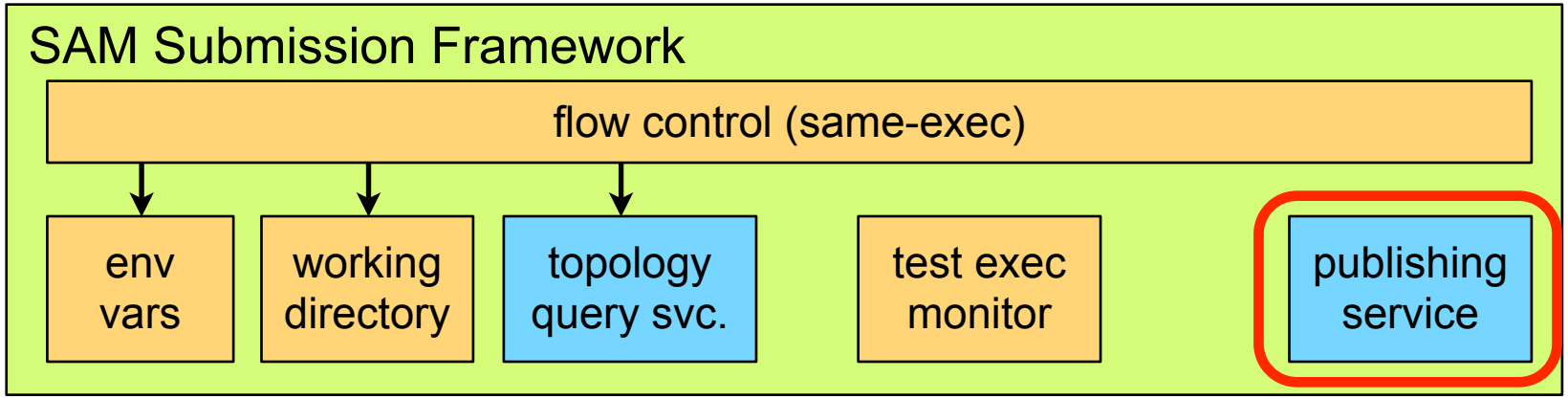
Topology query service



Topology query service

- Uses data model and DB shared by SAM and GridView:
  - sources of information: GOCDB, BDII, static lists (HEP experiments)
  - entities: region, country, tier, site, node/service instance, VO, maintenance records
  - “broken” tier representation: no VO dependency, no Tier-2 cloud representation
- Simplified query language introduced:
  - filter on attributes by list of values
  - relations handled automatically (graph theory algorithm)
- Database structure and business-logic went through intensive work (GOC DB vs. BDII correlation, etc.)
- Query service unchanged since beginning of SAM

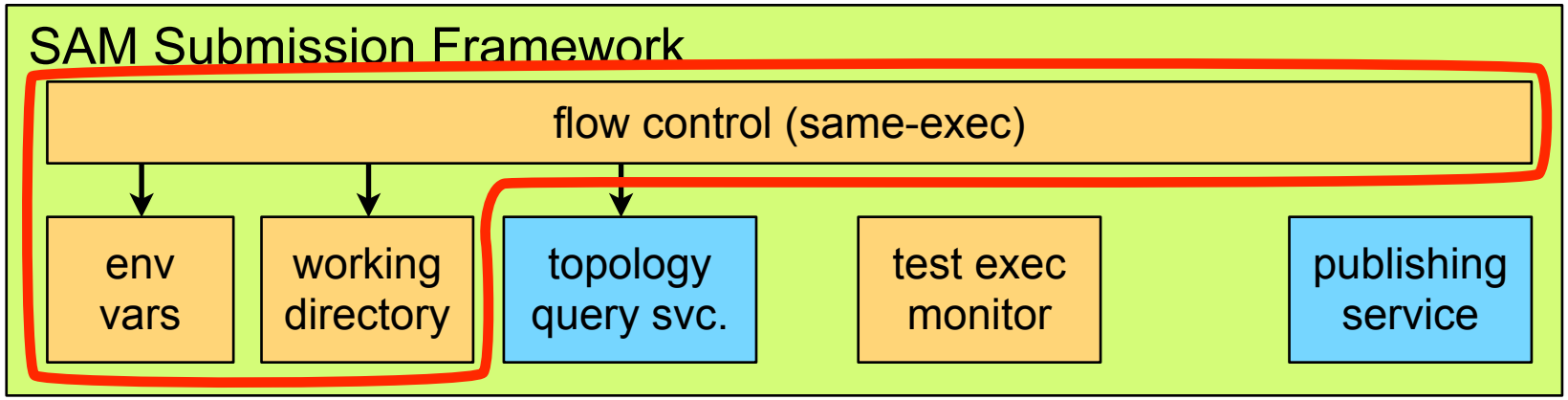
Publishing service



Publishing service

- Central web service receiving test results and storing them into the DB
- Portable client script (Python)
- Simple transport layer replacement to R-GMA:
  - better control over the transport (no data loss)
  - better aggregating capabilities (translation from transport schema to the normal form)
  - centralised and insecure (!) - but no major problems yet
- Server implemented by GridView
- Client implemented by SAM team
- Used by standalone sensors (GStat, etc.)
- Extended recently with JobWrapper tests tables
- Used by GridView in parallel to R-GMA for data transfers monitoring (their own client)

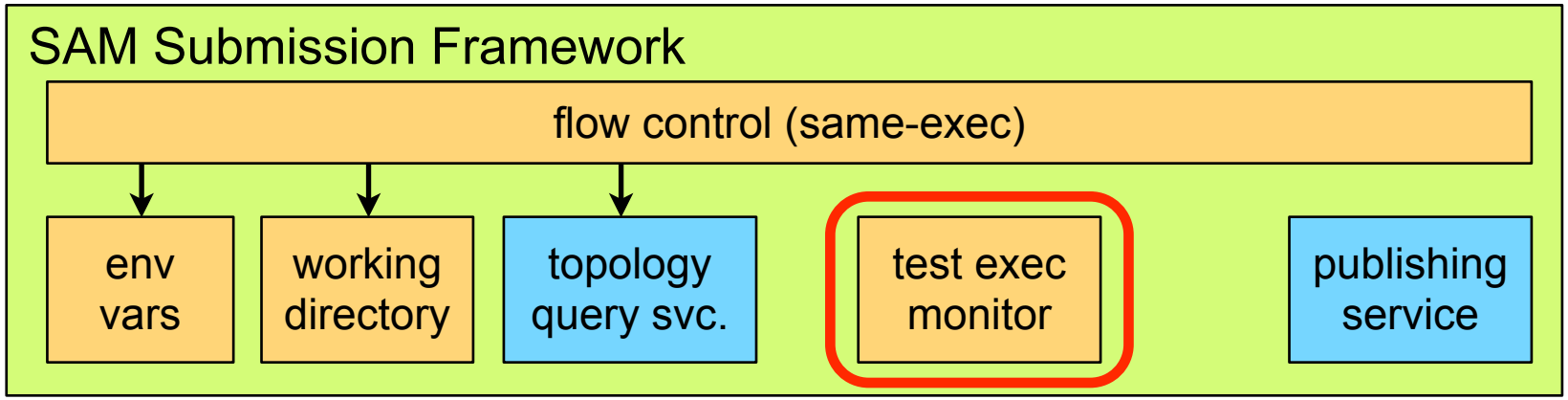
Workflow control



Workflow control

- Role:
  - Implements workflows defined by the sensor interface
  - Maintains sensors environment and state (working directories)
- Number of developments and bug fixes for workflows:
  - requirements from SAM Admin Pages
  - consistency of input parameters
  - per sensor locking (in validation)
- Backward compatibility between releases maintained

Test execution monitor



Test execution monitor

- A callback command for sensors to wrap executed tests:
  - catch and interpret STDOUT/STDERR and exit code
  - build test result output file - part of sensors interface
  - monitor the execution time, generate a test timeout error
- Few bug fixes related to timeout mechanism (catching as much of output as possible)

Open issues

- Convergence of sensors interface with probe standard from Monitoring WG
  - “wrapped” approach (both ways): 1-3 months
  - native usage of probe standard: >6 months
- Integration with Nagios (ongoing experiment):
  - main Nagios features:
    - automatic dependency handling
    - scheduling tests over time
  - timeline: 1-3 months
- Insecure and centralised publishing service
  - new Grid Publisher from Monitoring WG (6 months?)
- Topology query duplicates XML interface
  - Convergence plan needed to avoid duplication of topology query services (3 months)

Open issues (cont.)

- Design issues in sensor interface
  - too much logic in sensors (resolving dependencies, scheduling individual tests)
  - hard to catch full test output in case of timeout
  - solutions:
    - More lightweight sensors: automatic dependency handling within framework (3 months)
    - Using probe standard and integration with Nagios
- No automatic load balancing for tests scheduling (load peaks)
  - Integration with Nagios may help
  - Otherwise, development of scheduling module
    - timeline: 6 months