OPUS: a UWS client/server to access work clusters

Mathieu Servillat

Observatoire de Paris
Paris Astronomical Data Centre

IVOA Cape Town meeting

Computation at Observatoire de Paris

- **Tycho work cluster**
  - **16 nodes** : tycho[01-16]
    - 16 cores, Intel Xeon 2.60 GHz / 64 Go mem/node / 1,7 To disk space
  - **12 nodes** : quadri[17-28]
    - 8 cores, Intel Xeon 2.27 GHz / 24 Go mem/node / 160 Go disk space

- **Simple Linux Utility for Resource Management**
  - **Manage resources**
    - Job execution
    - Job limitations /node/user
    - Node extinction
  - **Job Scheduler**
    - Backfill, fairshare, priority, preemption

Job Management at PADC

- **Specific context**
  - Work cluster (Tycho)
  - Job scheduler (SLURM)
- **Needs for PADC projects**
  - **Web based clients**
    - Data processing jobs
    - Wrap simulation codes
  - **Interface** to computational resources
    - Using VO Universal Worker System

OPUS UWS server

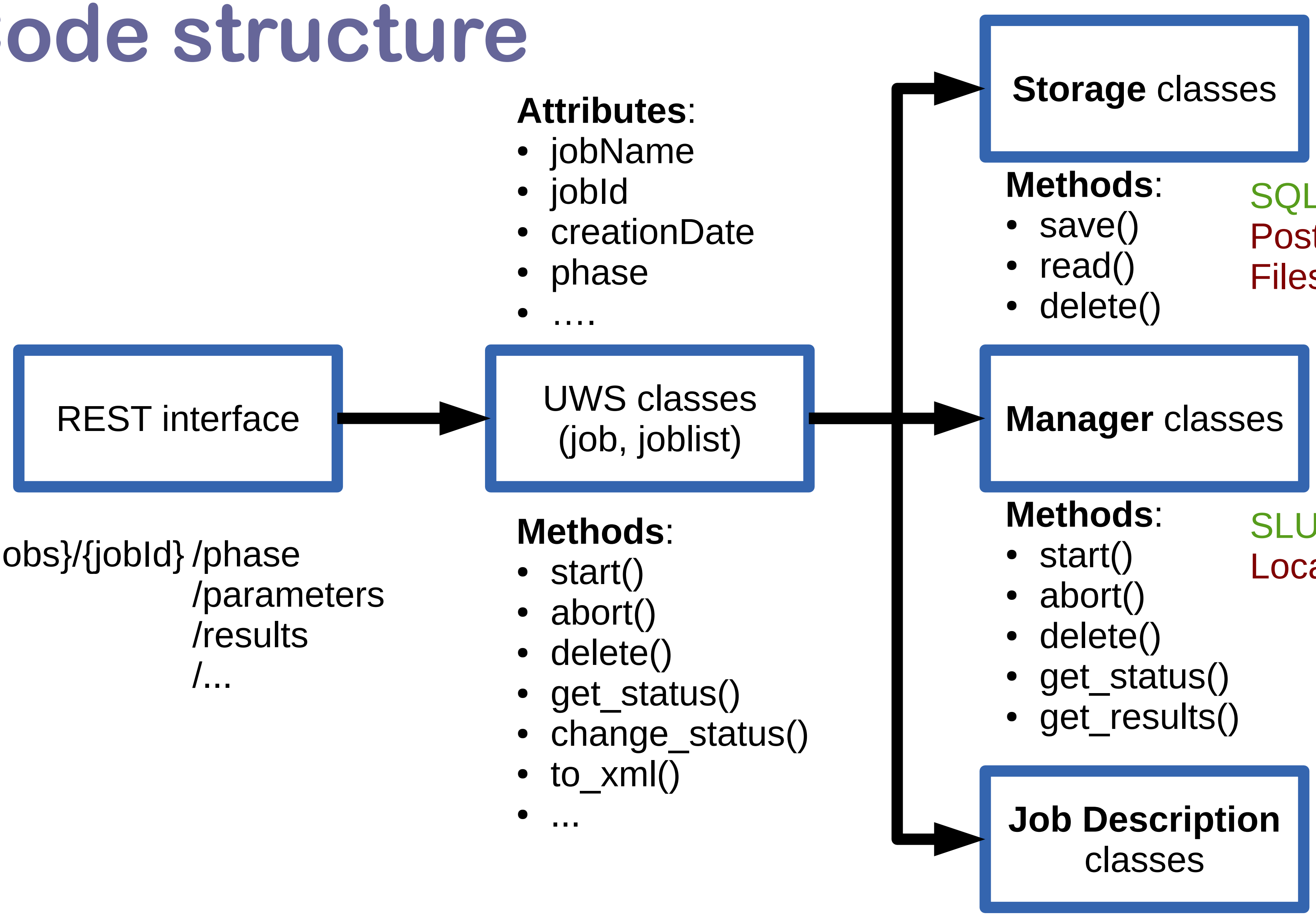
Main features

- IVOA **standard**
  - Universal Worker System (UWS)
- **REST** architecture
  - Python micro-framework: bottle.py
- **Collaborative** development
  - Git server at PADC (gitolite)
  - GitHub:
    https://github.com/ParisAstronomicalDataCentre/OPUS
    http://uws-server.readthedocs.org
- **Tests** and **quality**
  - Unit tests with `unittest` and `webtest`
  - Activity history with `logging`

Prototype available

https://voparis-uws-test.obspm.fr

Code structure



Attributes:
- jobName
- jobId
- creationDate
- phase
- ….

{jobs}/{jobId} /phase
/parameters
/results
/...

UWS classes — **Methods**:
- start()
- abort()
- delete()
- get_status()
- change_status()
- to_xml()
- ...

Storage classes — **Methods**:
- save()
- read()
- delete()

SQLite
PostgreSQL
Files

Manager classes — **Methods**:
- start()
- abort()
- delete()
- get_status()
- get_results()

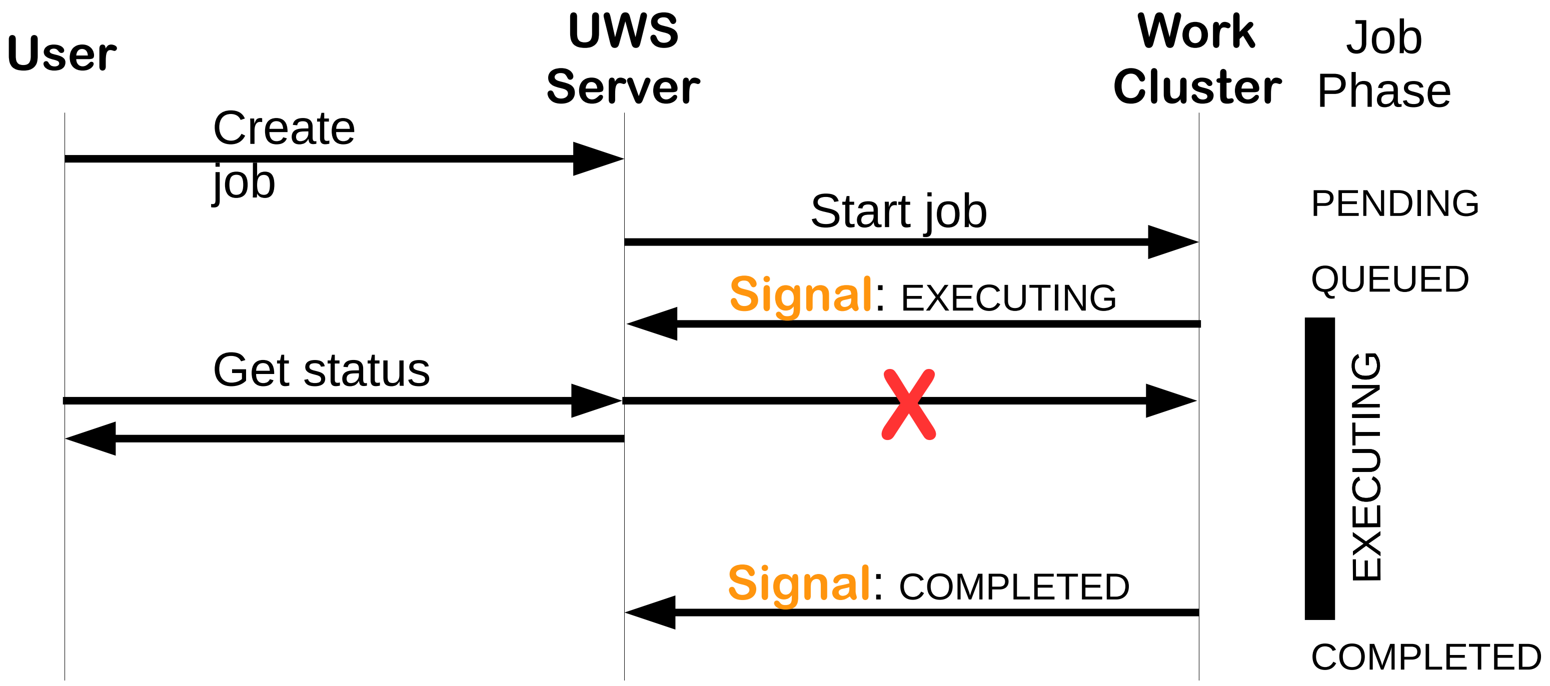
SLURM
Local

Job Description classes — **Methods**:
- save()
- read()

WADL
VOTable
PDL

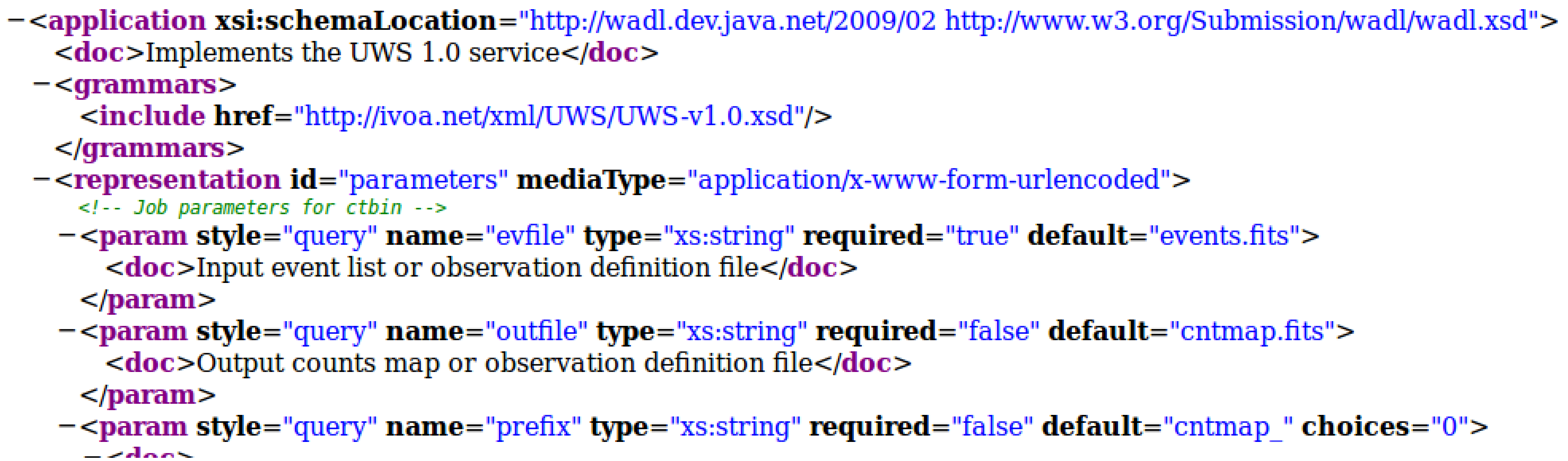
UWS server features

- **Separate job description from work cluster**
  - Wait for work cluster signals
  - Avoid (too many) status queries to work cluster



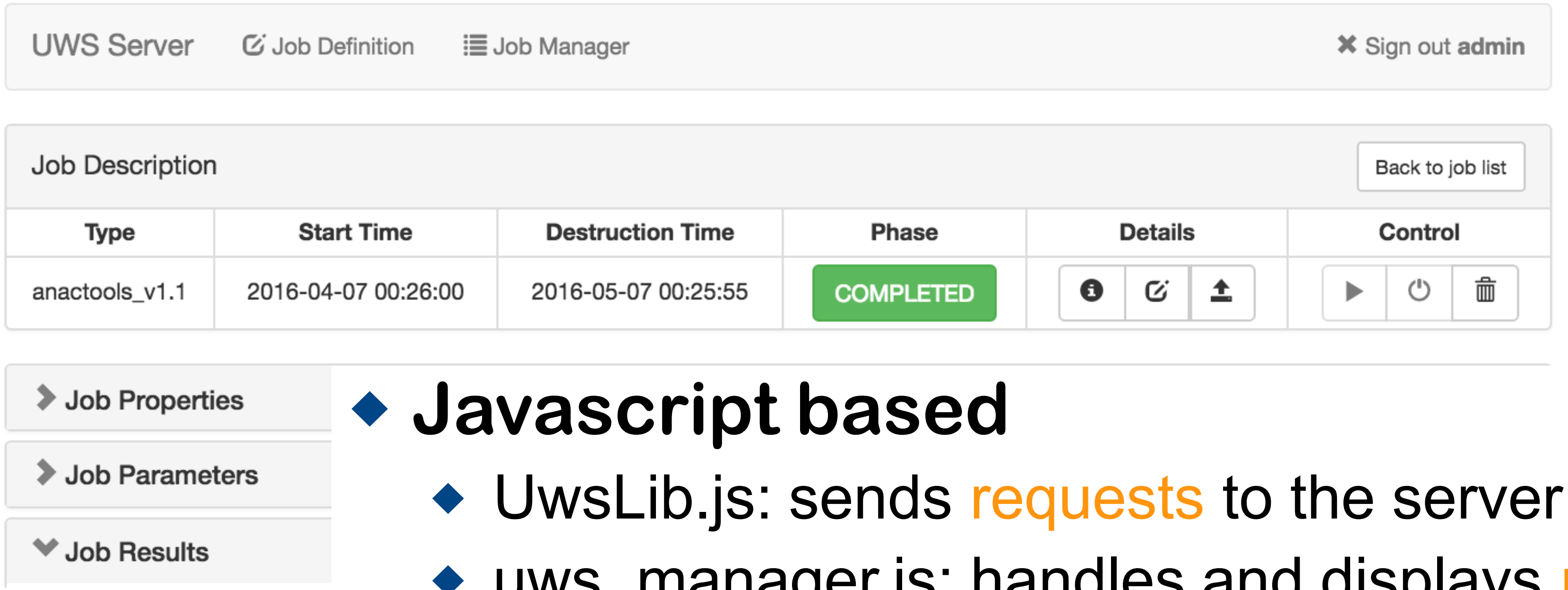
UWS server features

- **Full description of the UWS web service**
  - One **WADL** file (Web Application Description Language)
  - Describe parameters and results
  - Auto-generate parameter forms, results access
  - Test if submitted parameters are valid



```
-<application xsi:schemaLocation="http://wadl.dev.java.net/2009/02 http://www.w3.org/Submission/wadl/wadl.xsd">
   <doc>Implements the UWS 1.0 service</doc>
 -<grammars>
     <include href="http://ivoa.net/xml/UWS/UWS-v1.0.xsd"/>
   </grammars>
 -<representation id="parameters" mediaType="application/x-www-form-urlencoded">
     <!-- Job parameters for ctbin -->
   -<param style="query" name="evfile" type="xs:string" required="true" default="events.fits">
       <doc>Input event list or observation definition file</doc>
     </param>
   -<param style="query" name="outfile" type="xs:string" required="false" default="cntmap.fits">
       <doc>Output counts map or observation definition file</doc>
     </param>
   -<param style="query" name="prefix" type="xs:string" required="false" default="cntmap_" choices="0">
```

UWS client



Type	Start Time	Destruction Time	Phase	Details	Control
anactools_v1.1	2016-04-07 00:26:00	2016-05-07 00:25:55	COMPLETED		

- **Javascript based**
  - UwsLib.js: sends requests to the server
  - uws_manager.js: handles and displays responses
    - Integration with Bootstrap3
    - HTML page with specified <div> elements (id=joblist, parameters, results…)
- **Job definition editor**
  - Interface to create JDL file
  - Define parameters/results, bash script

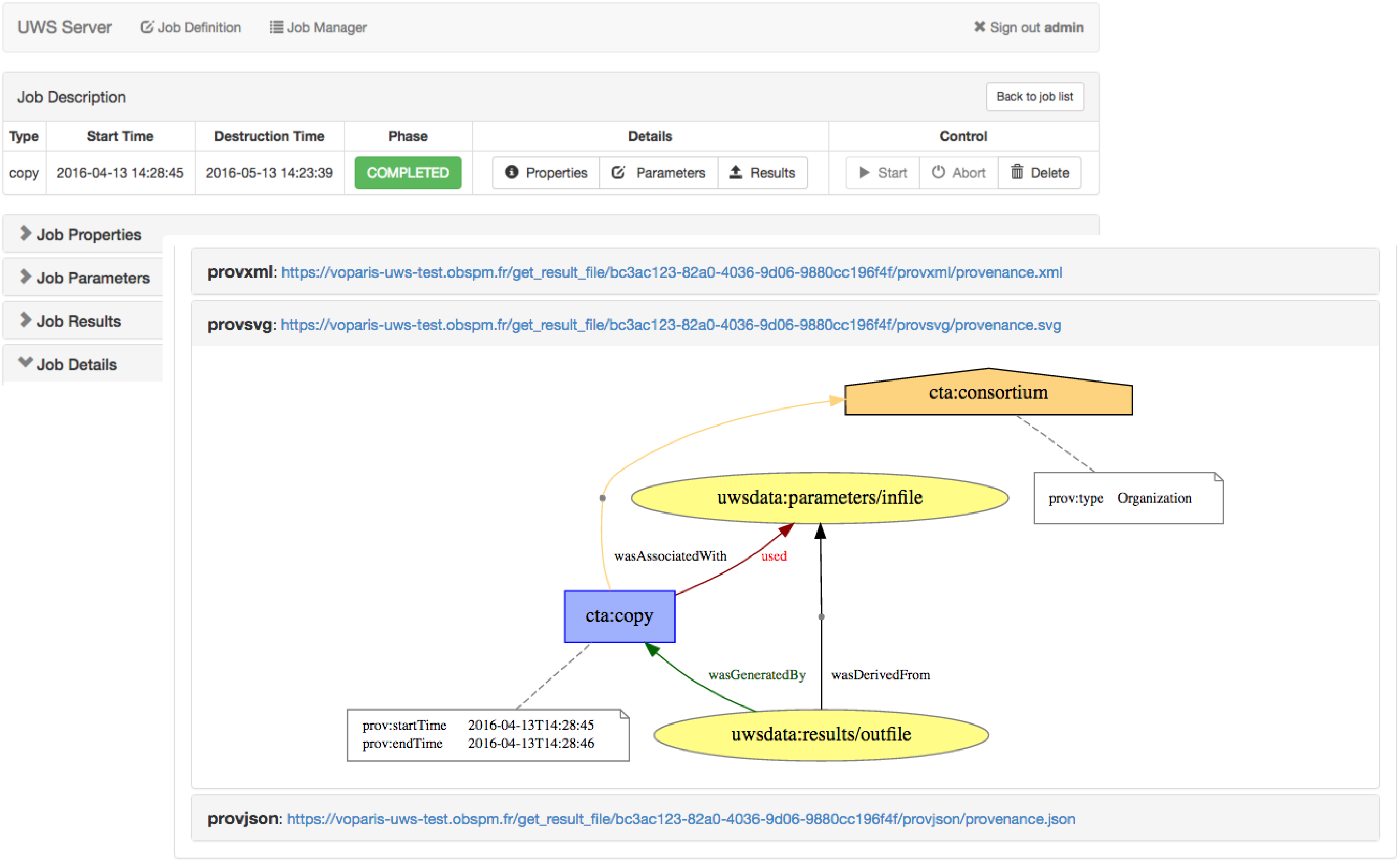
UWS Standard comments

- **Not used in the v1.0 implementation**
  - PENDING barely used
    (Client sends all parameters and starts job)
  - HELD not necessary (managed by SLURM)
  - SUSPENDED not yet included (managed by SLURM)
- **Some redundancy**
  - start, delete, set parameters
  - But easy to implement
- **To be implemented (v1.1 and more)**
  - Pagination
  - Filters by phase
  - WAIT= (though not critical in our case)
  - Authentication system (using SSO/Shibboleth or HTTP auth)
  - Connection with Provenance DM and DataLink ?

UWS Standard comments

- **If several jobs are defined**
  - {jobName}/{jobId}/…
  - But jobId is unique, no need to know jobName
- **Storing Job Definitions**
  - VOTable, using PARAM for parameters
  - PDL?
- **Job Definition Language**
  - Need more info, related to Provenance
  - Input entities ≠ parameters
  - Output entities ~ results, but those are URLs pointing to the entity

From UWS to Provenance



UWS Server | Job Definition | Job Manager | Sign out **admin**

Job Description | Back to job list

Type	Start Time	Destruction Time	Phase	Details	Control
copy	2016-04-13 14:28:45	2016-05-13 14:23:39	COMPLETED	Properties / Parameters / Results	Start / Abort / Delete

- Job Properties
- Job Parameters
- Job Results
- Job Details

provxml: https://voparis-uws-test.obspm.fr/get_result_file/bc3ac123-82a0-4036-9d06-9880cc196f4f/provxml/provenance.xml

provsvg: https://voparis-uws-test.obspm.fr/get_result_file/bc3ac123-82a0-4036-9d06-9880cc196f4f/provsvg/provenance.svg

provjson: https://voparis-uws-test.obspm.fr/get_result_file/bc3ac123-82a0-4036-9d06-9880cc196f4f/provjson/provenance.json