Middleware independent Information Service

Harry Enke,
Astrophysical Institute Potsdam
AstroGrid-D

Outline

1. Why middleware independent components?
2. Stellaris as middleware independent information service
   - Intro
   - RDF + SPARQL
   - Components
3. UseCases / Examples
   - Job information
   - Resource information
   - Integration of Resources into the Grid (Robotic Telescopes)
4. Summary

Thanks to M. Hoegqvist, I. Nickelt and the AstroGrid-D team

Why middleware independent components?

Many users and Virtual Organisations are using more than one middleware.

Communities have to broker agreements with providers using different middleware.

Information from different middleware components may not be consistent.

Interfacing with non grid-components may be necessary.

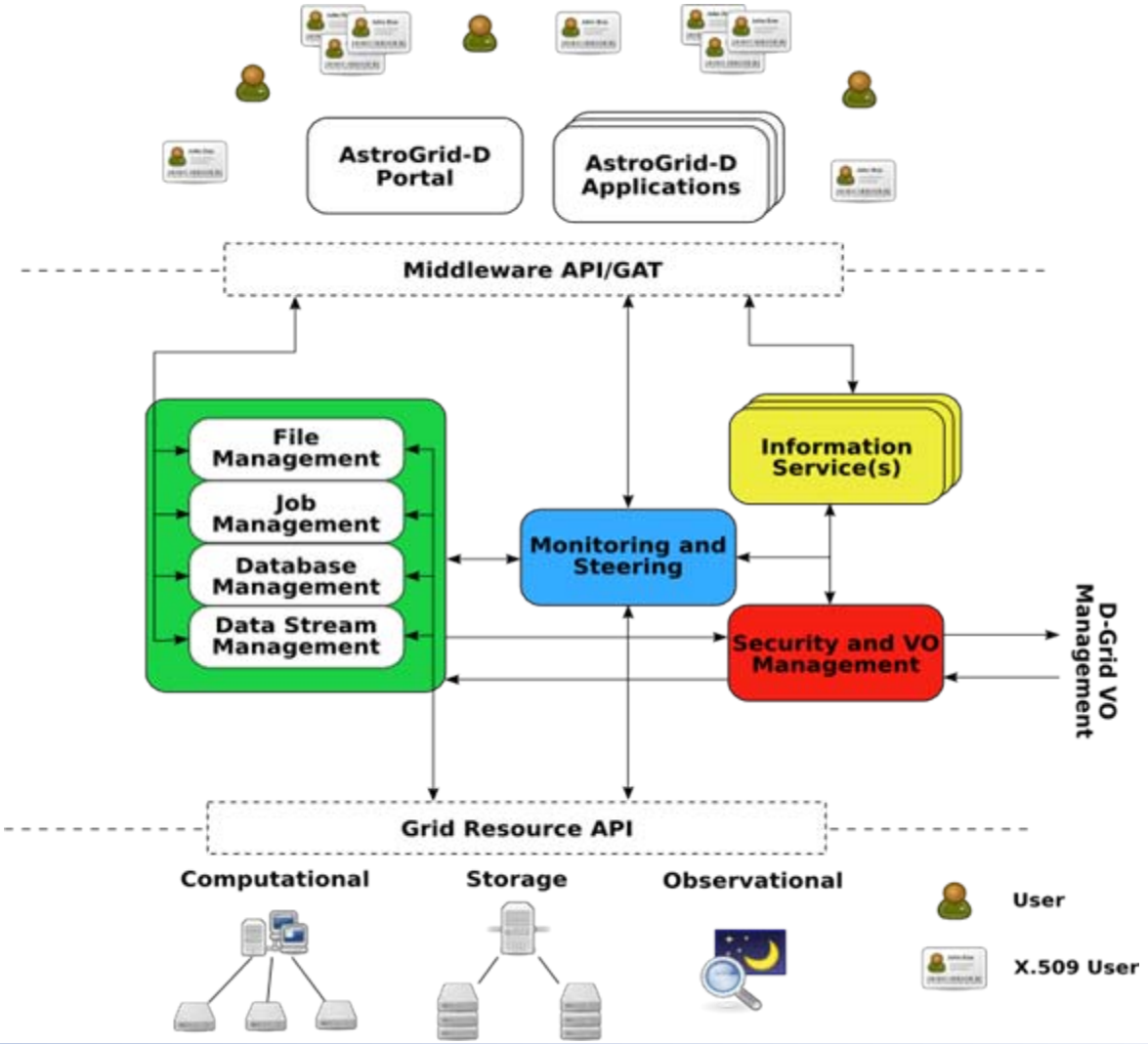
Why middleware independent components?

Both user and VOs want

- information on available resources or
  brokering facilities
  for storage and compute requirements
- information about charges (fees)
- information about their jobs
  (runtime and accounting)
- access and store metadata for their data sets

in a middleware-independent form.

Metadata Requirements



Metadata Requirements for a Community Grid

- Resource metadata
- Activity State
- Application metadata
- Scientific metadata

Requirements for a CG information service

AstroGrid-D's requirements for Stellaris

- A uniform interface compatible with existing tools, based on standards
- support for flexible and extensible metadata schemes,
- integration of the following metadata types:
  - Resource metadata
  - Activity State
  - Application metadata
  - Scientific metadata
- authentication and authorization for access control.

The Stellaris Information Service

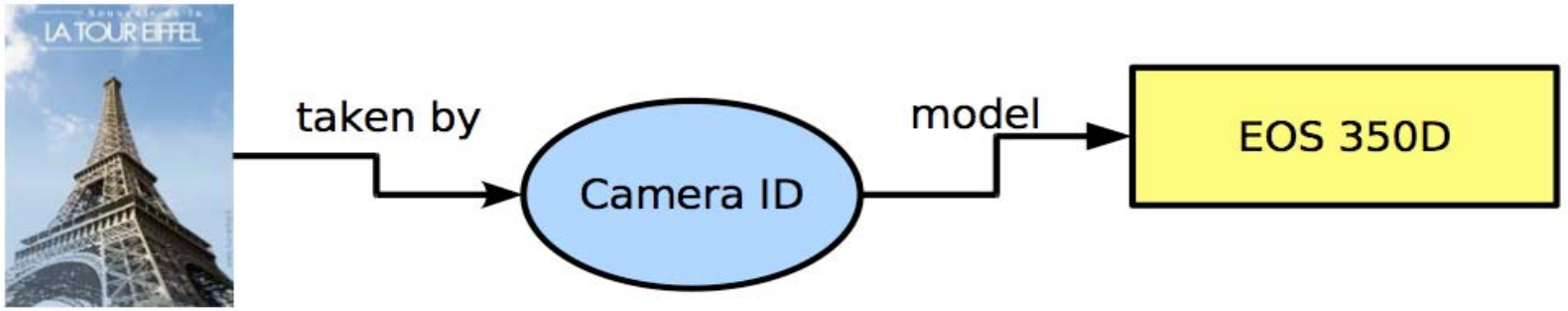
Stellaris is a database application

- Data is represented in RDF format
- Data is uploaded and retrieved via web based protocols (http(s), soap)
- SPARQL is a query language for data extraction

  (SPARQL Protocol and RDF Query Language is a recursive acronym)
- Fine grained access control is possible through grid security mechanisms
- Persistent storage on disk by RDBMS

Stellaris: RDF example

A simple RDF tuple (graph)



subject predicate object

Stellaris: RDF Introduction

RDF is a recommendation of W3C. Numerous tools and interfaces available from the semantic web community.

Stellaris uses RDF/XML syntax.

- A general XSLT from XML to RDF is available.
- Stellaris does translation on the fly.

The RDF data model makes statements about resources.

A statement is divided into the

- resource (subject) itself, a
- predicate describes an unidirectional relation to an
- object

This (subject, predicate, object)-tuple is often called a RDF triple.

Stellaris: RDF Introduction

Subject and predicate are URI (Uniform Resource Identifiers), while the object can be literal (a value) or an URI.

RDF-URI are globally unique identifiers.

Combining multiple RDF-Statements generates RDF-Graphs or contexts.

Contexts are organized as tree-like structures within Stellaris.

Besides from RDF/XML syntax the Notation 3 (N3) format is supported as well.

RDF vs XML

- XML is a document format

  A XML document can be interpreted in many different ways

- RDF is an information model

  RDF can be represented in an XML format

  A RDF document has only one interpretation, independent of the representation

Stellaris: Vocabulary / Schema

- RDF is semi-structured data, which allows for easy changes of vocabulary without changing the stored data
- Vocabulary used with Stellaris:

  Well known XML schemata

  Dublin Core, GLUE

  Additional: custom vocabulary defined by community, VO, application or user

  RTML: Robotic Telescope Markup Language

Stellaris: SPARQL example

Photographer

"Alice"

Creation-date

Phone number

2003.06.05

555-444

"What is the name and phone number of the photographer who took the picture of the Eiffel tower?"

Input graph

```
SELECT ?phone_number ?name WHERE
{"Picture of Eiffel tower" "Photographer" ?name .
 ?name "Phone number" ?phone_number }
```

Number	Name
555-444	Alice

Output results

Stellaris: SPARQL query

Create an object (referencing various schema)

```
@prefix file: <http://www.gac-grid.de/schema/files#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rtml: <http://www.rtml.org/v3.1a#>
<http://storage.gac-grid.de/test/eaglenebula.fits>
      rdf:type <file:DataObject>;
          file:owner "Telescope user";
          file:location <http://telescopes.aip.de/pictures/eaglenebula.fits>;
          file:filesize "259342";
      rtml:Telescope <rtml://de.aip.Robotel/STELLA> .
```

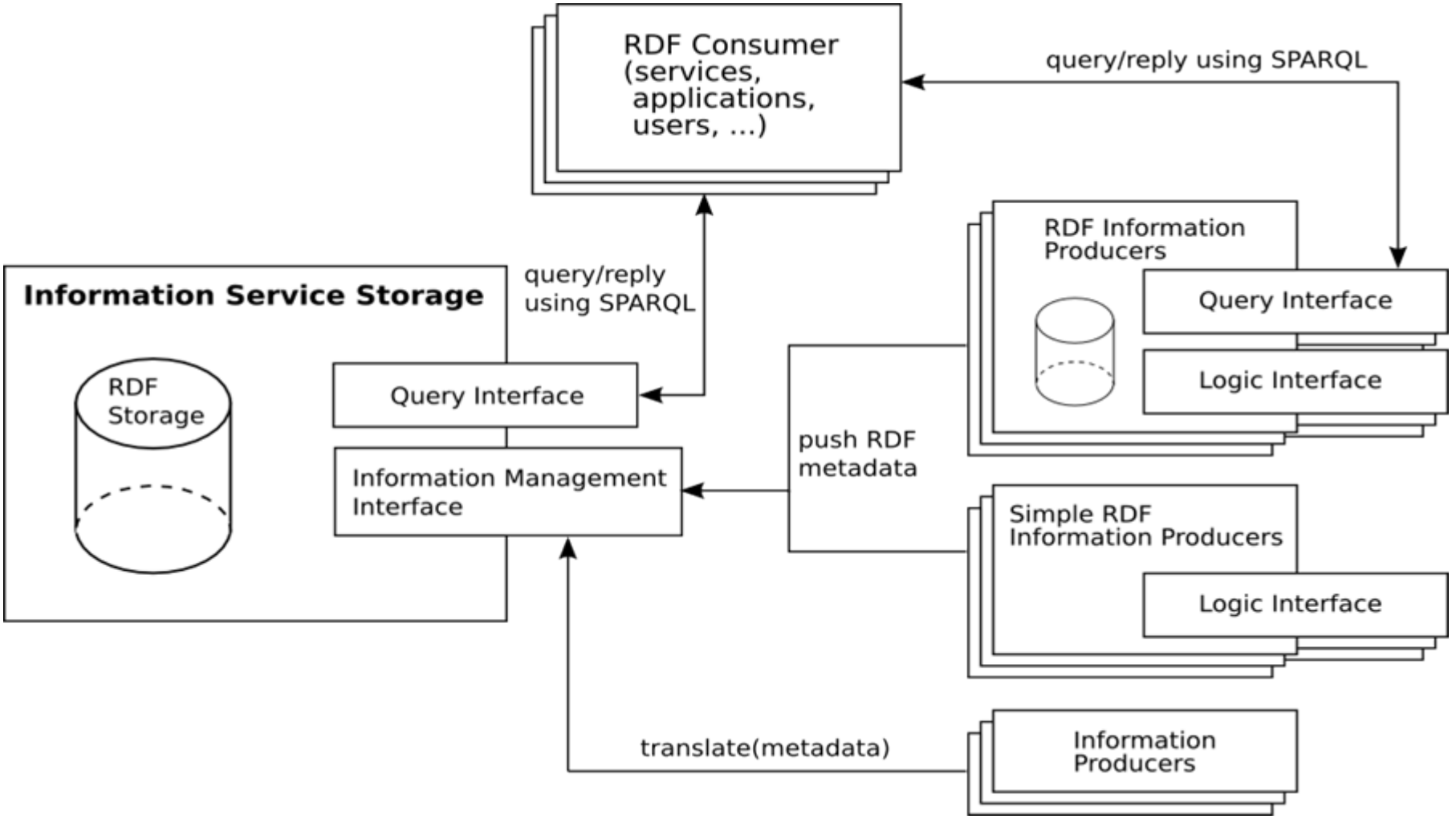
Stellaris: SPARQL query

Do a query:

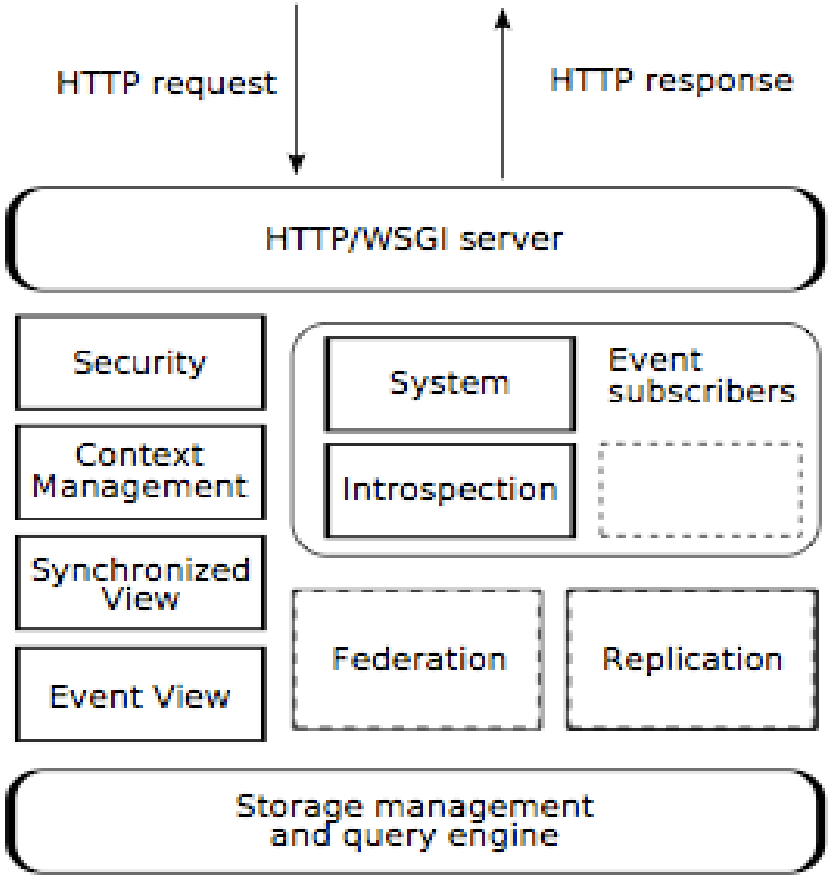
(return location, owner and telescope of the file referenced by URI)

```
PREFIX file: <http://www.gac-grid.de/schema/files#> .
PREFIX rtml: <http://www.rtml.org/v3.1a#> .
SELECT ?location ?owner ?telescope
FROM <http://telescopes.aip.de/context/robotic_telescopes>
FROM NAMED <http://stellaris.astrogrid.net/context/files>
WHERE {
    <http://storage.gac-grid.org/test/eaglenebula.fits> file:location ?location;
    file:owner ?owner; rtml:telescope ?telescope .
}
```

Stellaris: Internals



Stellaris: Interface



Request-Handler:

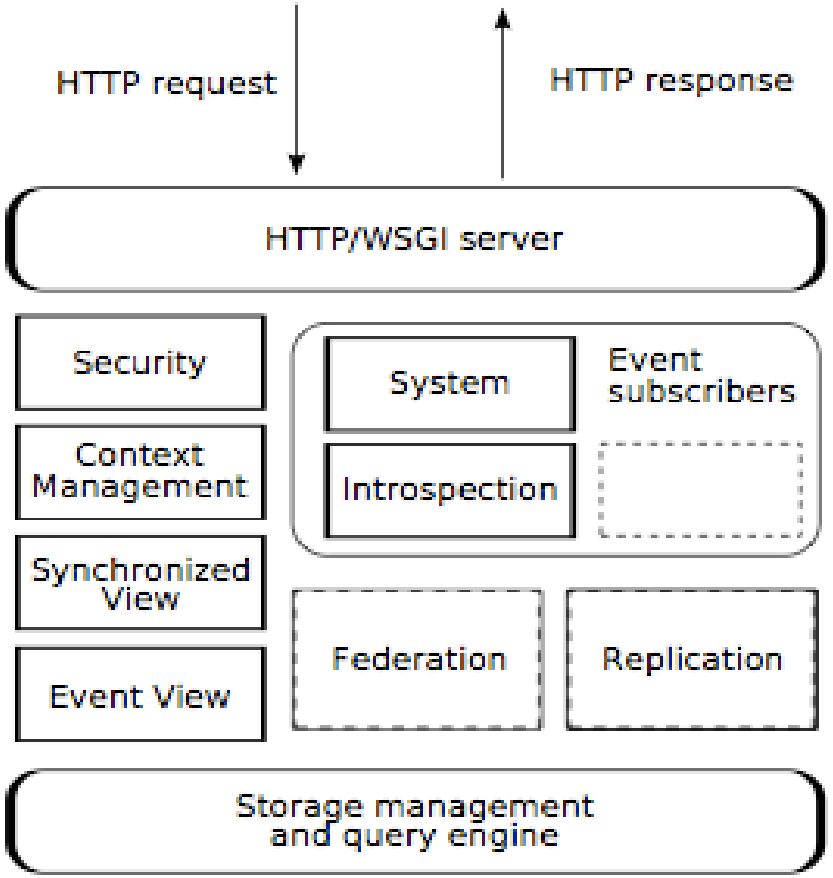
Security:

Handles authentication and authorization (ACL, VOMRS, X509)

Context Management:

internal interface for commands:
create / retrieve / update / delete
query

Stellaris: Interface



Request-Handler:

Synchronized View:

handles concurrency issues

Event View:

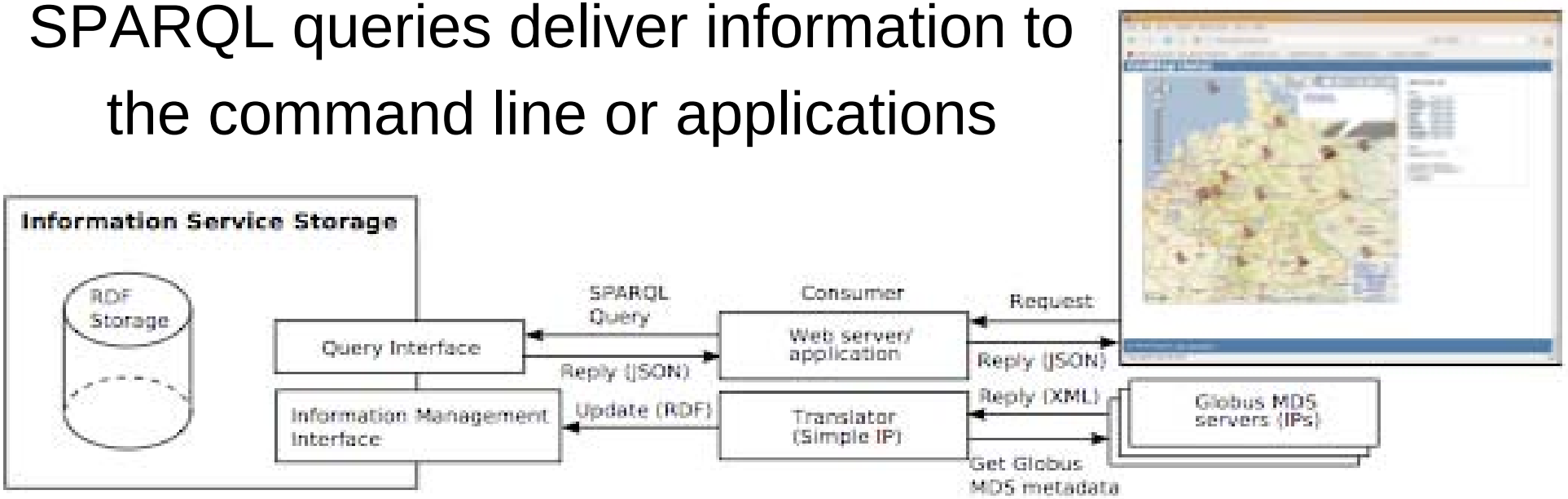
passes request to storage management or query engine

Resource Information with Stellaris

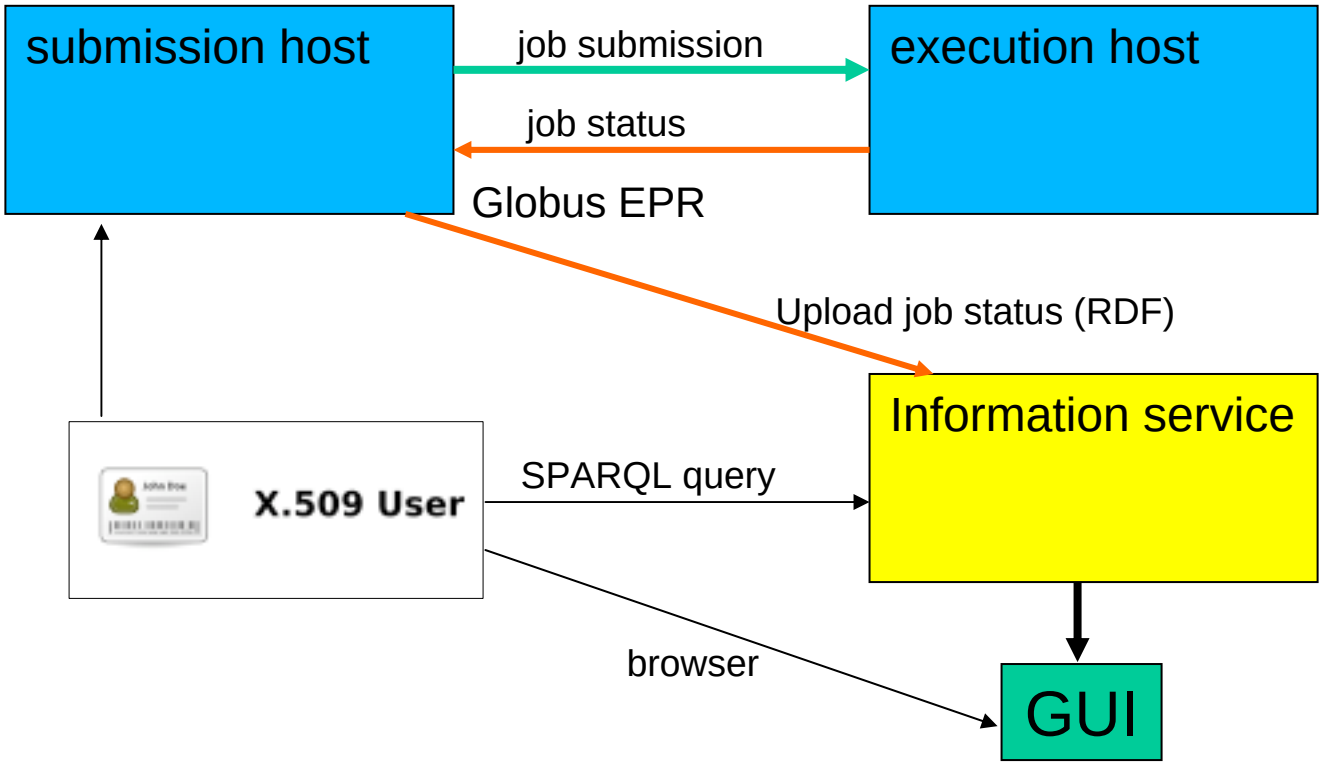
Globus-MDS information is uploaded to Stellaris after a XML to RDF transformation (GLUE).

A visual map displays current status data of resources

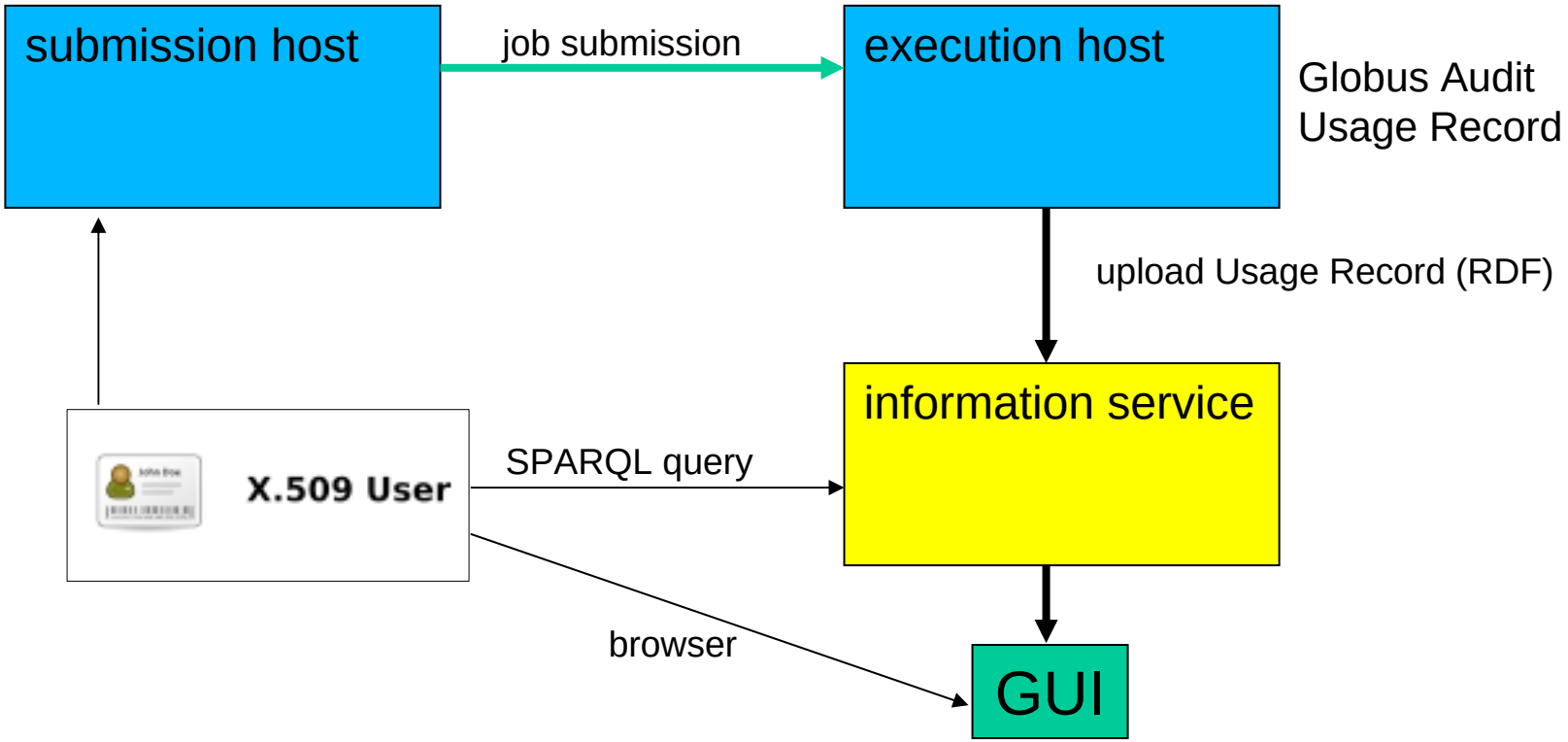
SPARQL queries deliver information to the command line or applications



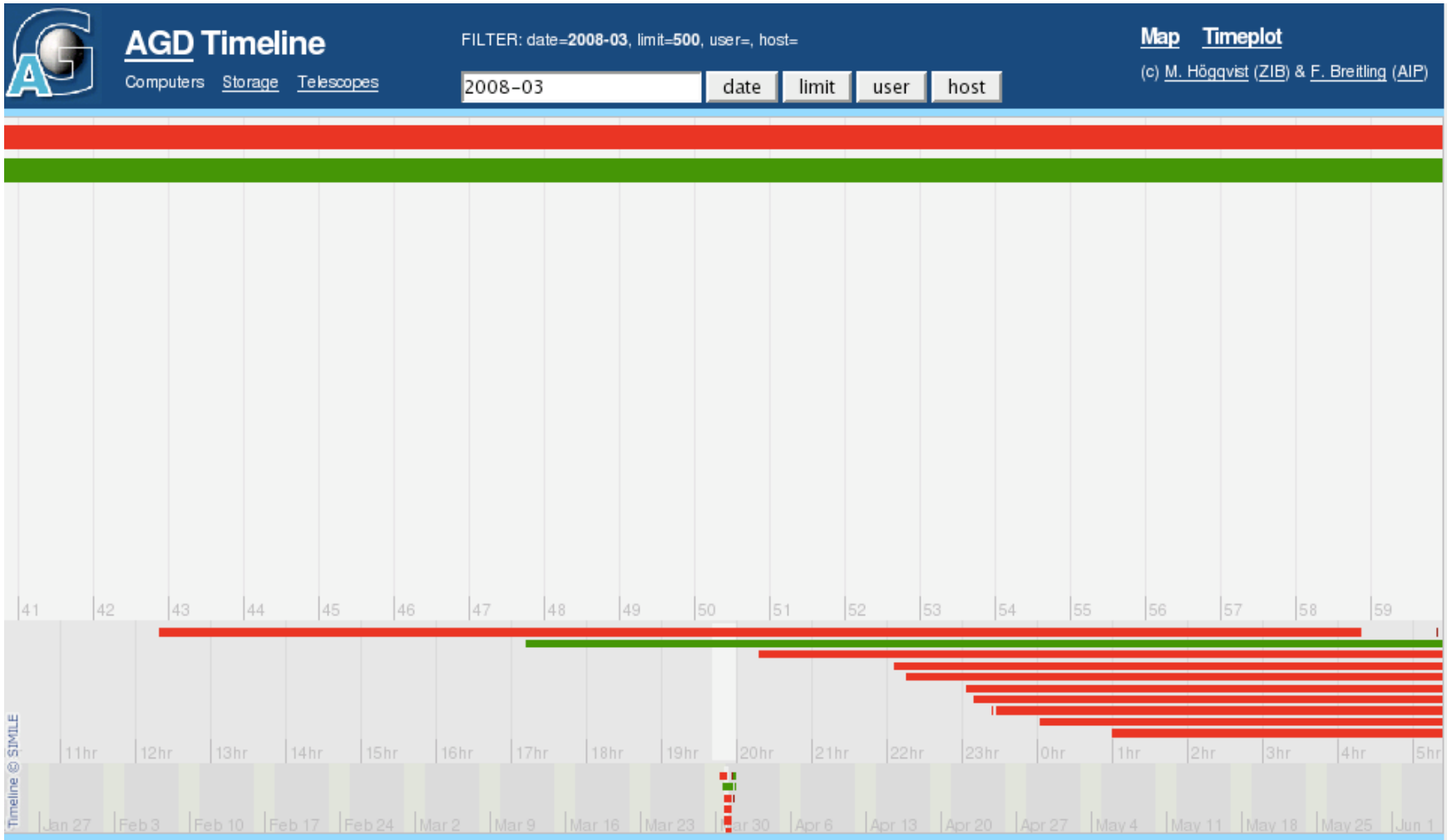
UseCases: Grid Job Monitoring I



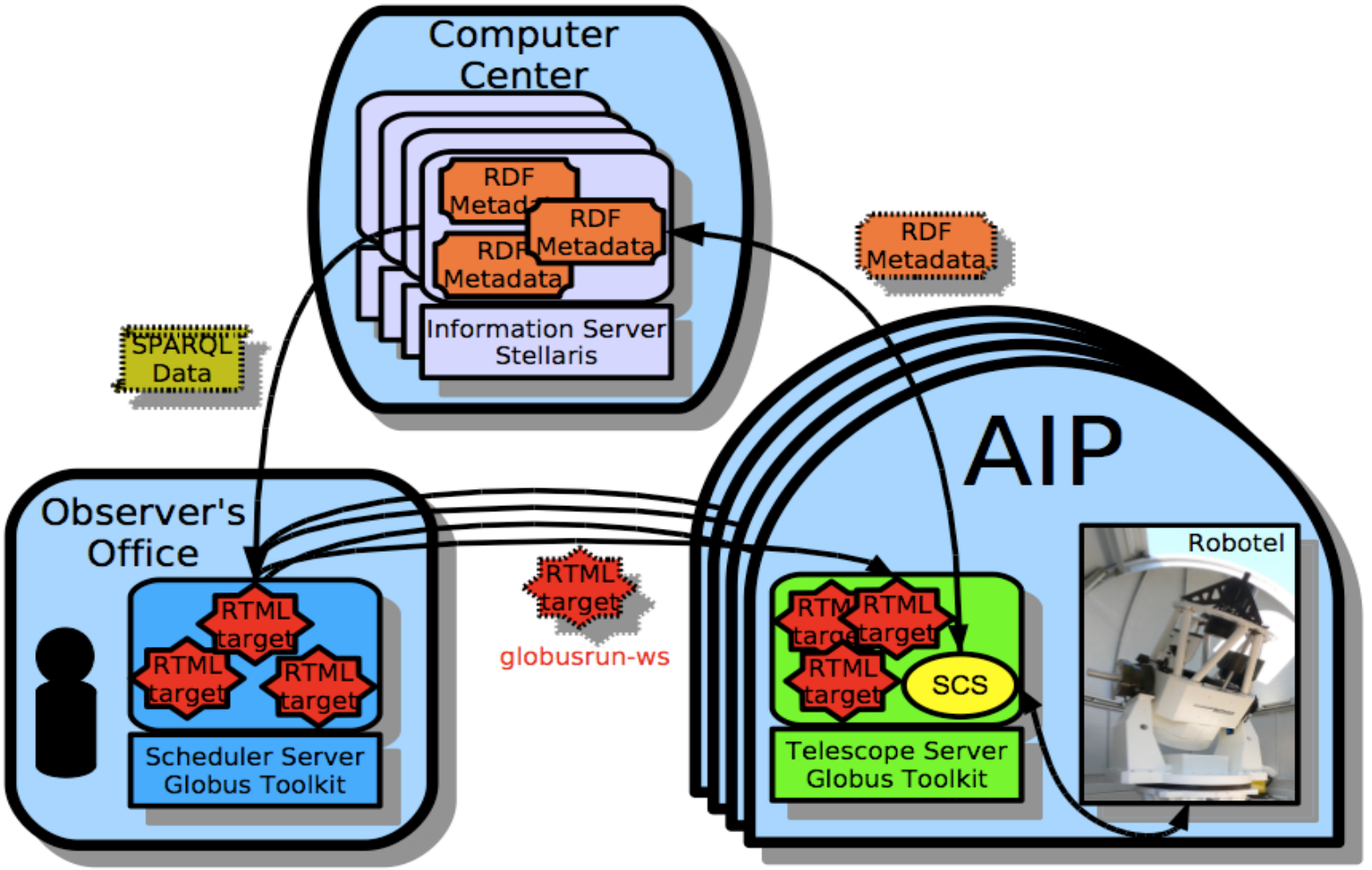
UseCases: Grid Job Monitoring I I



Grid Job Monitoring GUI Timeline



UseCases: Robotic Telescopes in Grid



Robotic Telescopes in Grid

Integration of robotic telescopes into the grid is an example of interoperability with other components

Basic middleware: Globus

for job submission, staging, job control

security,VO management

Robotic Telescope Components:

Controller host with interface for RTML requests

Heterogeneous Telescopes Network Component:

RTML

Interaction of components through Stellaris

Summary

Interoperability for user and VOs requires

an information service for middleware independent

(Resource metadata, Activity State information, Application metadata Scientific metadata)

Stellaris with the RDF data model and SPARQL query language is a good candidate for the task.

An information service such as Stellaris is needed for keeping the user or VO in the focus of the efforts to build an efficient eScience infrastructure