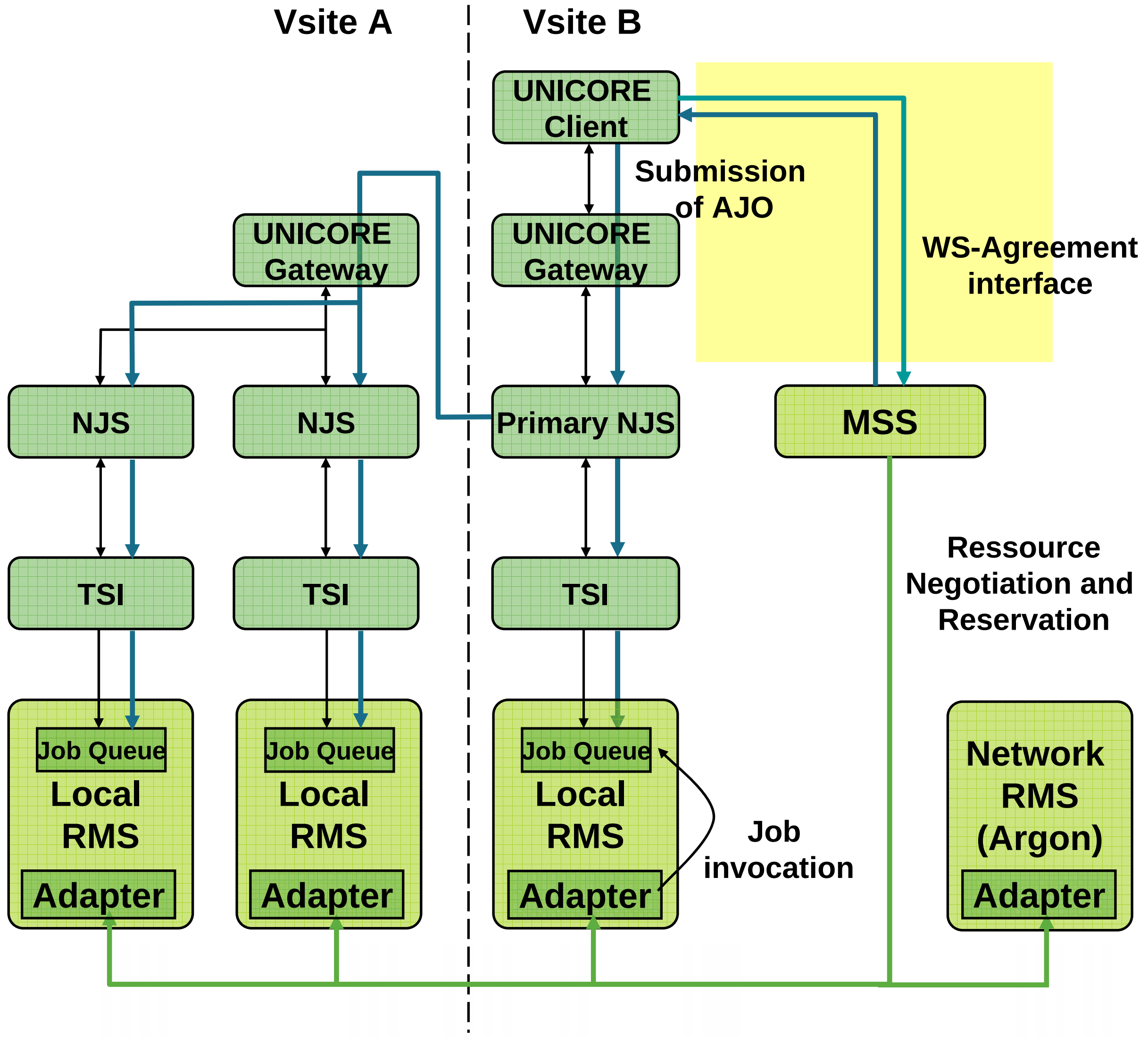
VIOLA WS-Agreement implementation

Oliver Waeldrich

GRAAP-WG

7. May, 2007 (GGF20 in Manchester)

VIOLA MSS - UNICORE Integration



- UNICORE Client sends request to MetaScheduler (WS-Agreement)
- MetaScheduler negotiates earliest time to run this job, requests the reservation of the requested resources and returns the WS-Agreement with additional Status, ID
- UNICORE Client creates Abstract Job Object (AJO) and sends it to the Primary Network Job Supervisor (NJS)
- NJS incarnates the AJO according to the information in the AJO and the UIDB, forwards it to the local Target System Interface (TSI) and sends the AJO to all other NJSs
- TSI creates the entry for the Meta-Job in the UNICORE Job Queue, and stores the job data in the User-directory
- Scheduler triggers job at start time

Issues of the VIOLA Implementation

- WS-Agreement Layer
  - Technical problems
    - We used Axis 1 as SOAP engine
    - No WSRF engine available
    - Problems with generating stubs and skeletons

      Patches of WS-Agreement specification necessary
  - Non-technical problems
    - We did not use the full power WS-Agreement
    - Proprietary protocols for co-allocation of resources
    - Missing negotiation capabilities

- Grid Middleware Layer
  - No reservation support in UNICORE 5 (out of the box)
  - Bypassing the Middleware
  - Limited security mechanisms

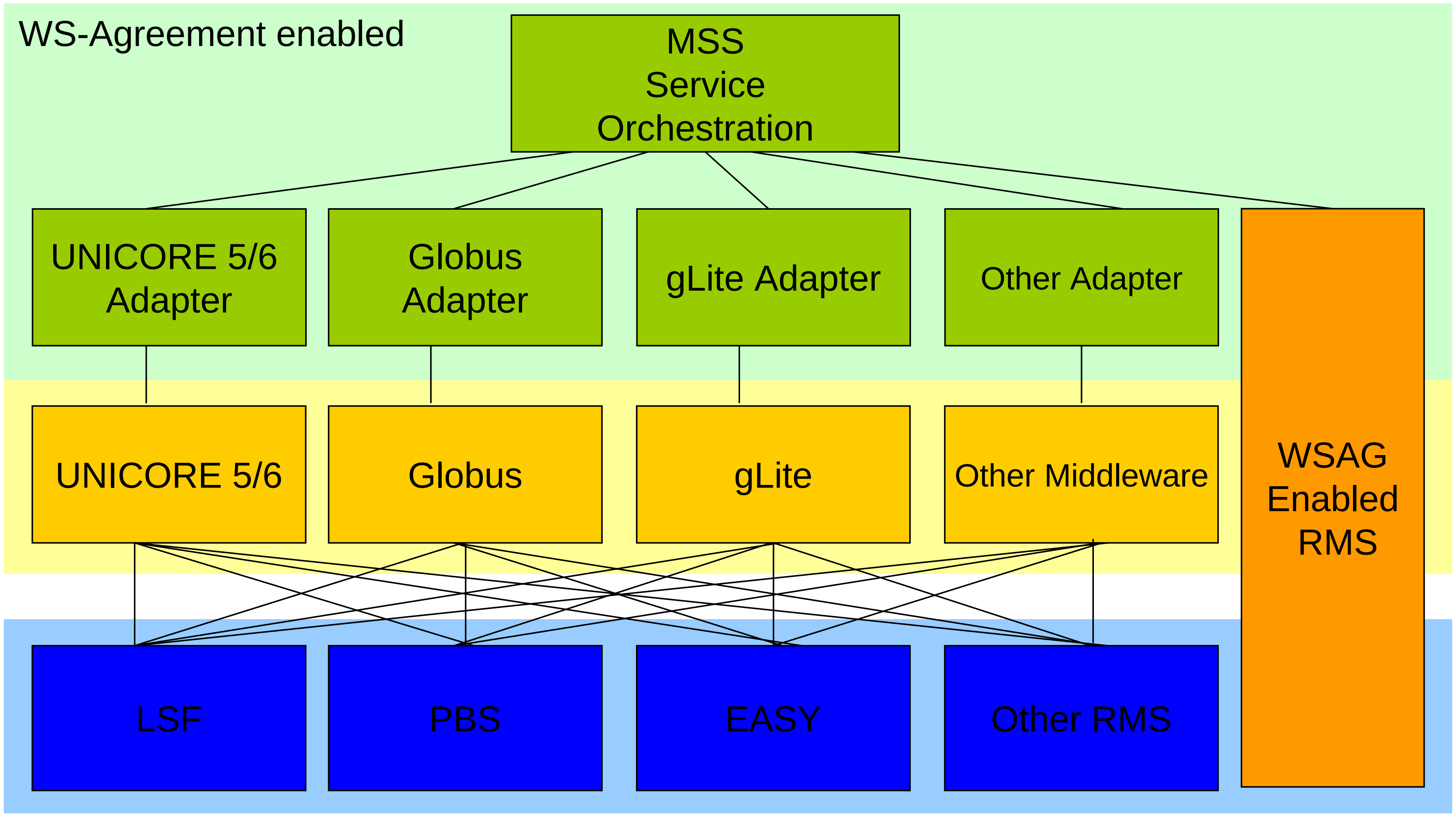
Goals for the MSS implementation

- Strong orientation on WS-Agreement standard
  - Service Level Agreements are key asset in Grid resource management
  - WS-Agreement is the standard the express SLA

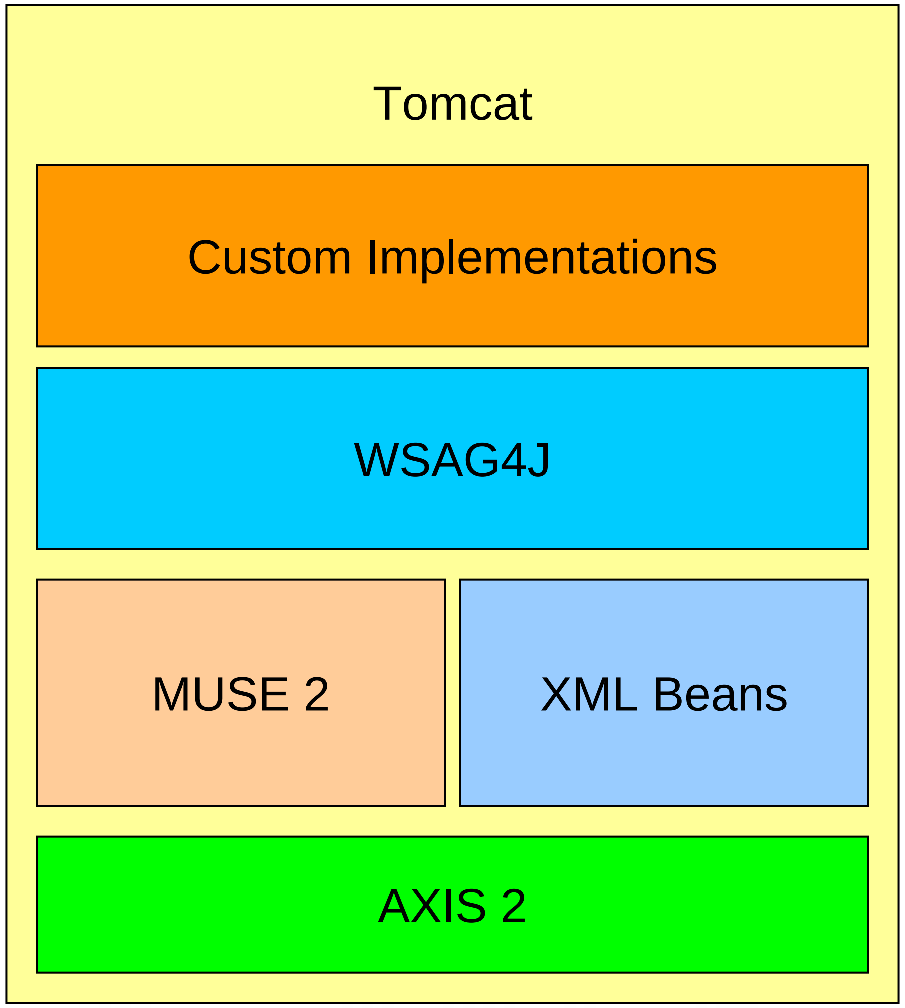
- Support of the base technologies
  - WSA, WSRF, WSN
  - Key to interoperability (but which is actually used?)

- Integration point for different Grid middleware
  - Stay decoupled from specific Grid middleware solution
  - Extension mechanisms for new middleware systems

Resource Orchestration using MSS



WSAG4J Hosting Environment



- ***Axis 2***
  - SOAP engine
- ***MUSE 2***
  - WSRF/WSN/WSDM engine
- ***XML Beans 2.1***
  - XML data binding
- ***Tomcat 3.x and higher***
  - Servlet engine
- ***Distribution***
  - Web application archive

Implementation State

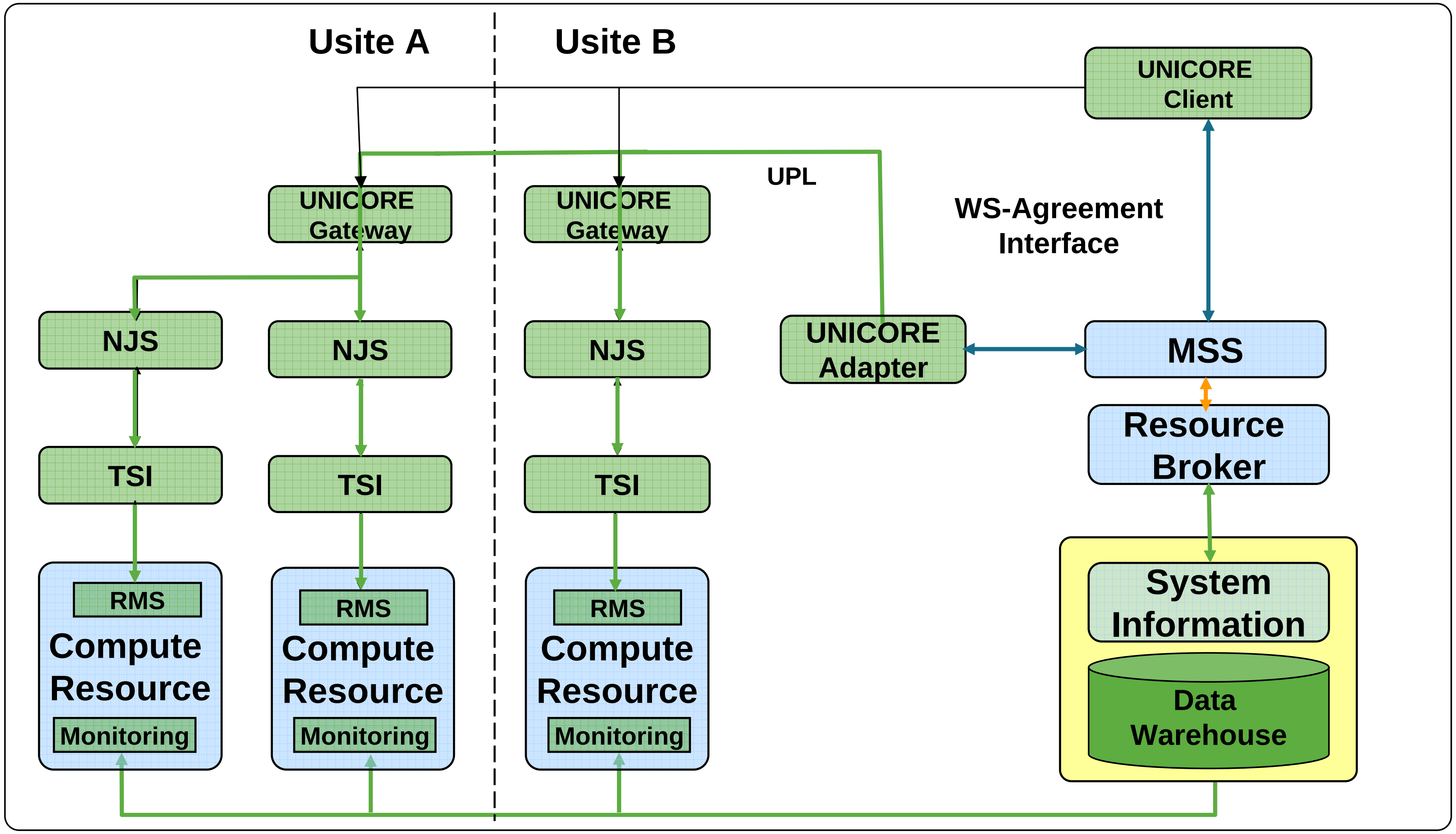
- ***Implemented WS-Agreement Version***
  - Final draft (submitted to editor)

- ***Implemented Port Types***
  - Agreement Factory
  - Agreement
  - Agreement State

- ***Implemented Features***
  - certificate based client authentication (TLS)
  - client side (register authentication handler with a factory instance)
  - server side (provide access to user certificates)
  - guarantee handling

- ***Still Missing***
  - Pending Agreement Factory
  - Agreement Acceptance
  - Separate client distribution (available soon)

WSAG4J
MSS UNICORE Integration



Current Work
WS-Agreement Processing

- ***Agreements must be processed by a service***
  - *Interpretation problem*
    - Service description, properties and guarantees must be evaluated and processed in a convenient way
    - *Trade of between specification and practice*

- **Agreements are or will be electronic contracts**
  - *Validation Problem:*
    - Only Agreements based on a valid template may be created.
    - Only Agreements are created, where all aspects are known and understood.
    - Completeness of the validation must be guaranteed

  - ***Restriction: This problem can be only solved on the WS-Agreement layer.***

→ First implementation available.

Conclusion

Overview of WS-Agreement related activities

- Motivation
- History
- Current state

- Usage of WS-Agreement
  - Resource Orchestration using UNICORE 5
  - Support for UNICORE 6 and GT4 (work in progress)
  - Agreement processing
- MSS and WSAG4J and processor are available at:

  http://packcs-e0.scai.fraunhofer.de/mss-project

  http://packcs-e0.scai.fraunhofer.de/mss-project/wsag4j