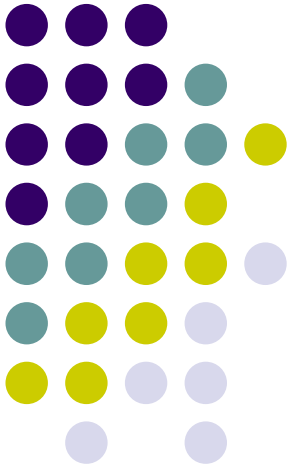
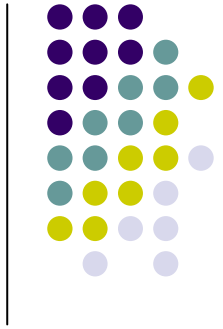
Towards GLUE Schema 2.0

Sergio Andreozzi
INFN-CNAF
Bologna, Italy

sergio.andreozzi@cnaf.infn.it

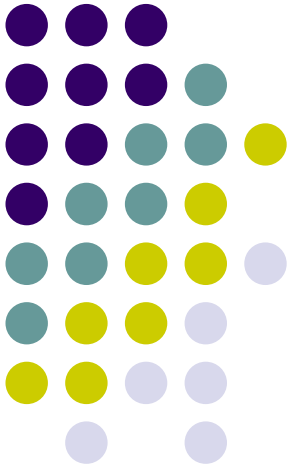


OUTLINE

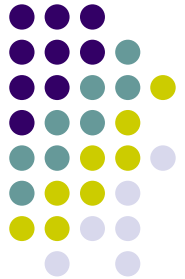


- GLUE Schema 2
  - Modeling guidelines
  - Relationship with JSDL
  - Relationship with CIM

Modeling Guidelines

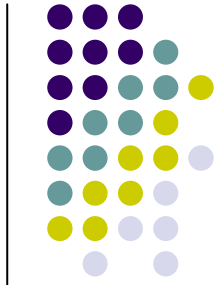


Generalization



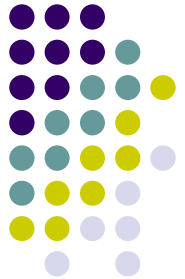
- Generalization based on resource type
  - e.g.: the query to search for storage space should not depend on the storage system type
  - The model should accomodate different storage systems, from simple disk-servers to complex SRM-managed systems

Contract-based Design



- Resource provision is based on agreements between providers and consumers
- The same contract can be implemented using different implementation choices
- The different implementation choices should not be exposed

Contract-based design - example



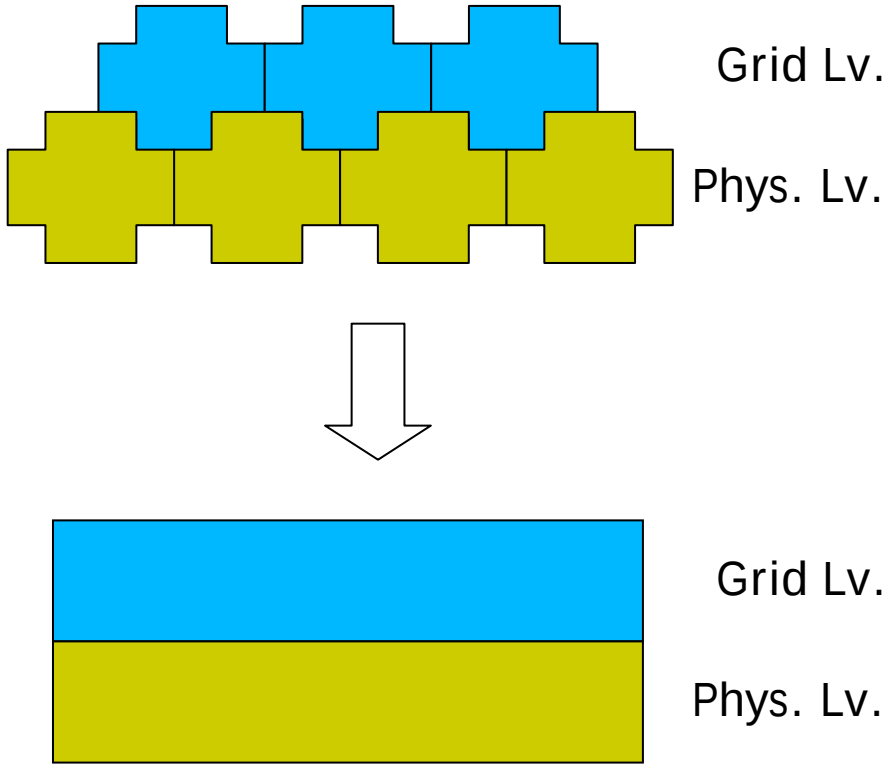
- A cluster with 100 CPUs provides 50% of share to VO A and 50% of share to VO B
- Two example configurations:
  - Config 1:
    - one batch queue with two different shares assigned to two different group ID;
    - Map each VO user to the related group ID
  - Config 2:
    - two batch queues, the share is associated to each queue
    - Each queue is authorized to one VO
- In GLUE Schema 1.2, the two scenarios lead to two different representations in the information service
  - 1 CE + 2 VOView vs. 2 CE

Contract-based design - example

- The same service is provided using two different configuration strategies
- Why two different representations at the Grid level?
- Contract-based view instead of implementation-based view

Contract-based design - comments

- Refine CE concept concentring on:
  - Isolation of physical level from Grid level
  - Identification of the core concepts for the Grid level
    - What is the shared resource?
      - Execution environment
    - How the access is policied?
      - Service differentiation



Denormalization/Redundancy

- Information service is read-only from the consumer side
- Redundancy can easy the query process
- Avoid many-to-many relationship as much as possible possible
- Always evaluate the impact of modeling choices on the query process

Heterogeneous Clusters

- Since the beginning, the GLUE Schema enables to model subsets of homogenous nodes in a cluster
- This feature has been never used because from the Grid level it is not possible to submit a job providing requirements on the worker node characteristics (at least with GT2 GRAM)
- This implied the usage of only one subcluster instance to be related to the less powerful machines!!!
- How can we better deal with this situation?
  - One single description with min/avg/max values?
  - A default description + the different descriptions?
  - Waiting for more powerful job submission protocols?

Multiple Endpoints to the Same Service

- The current schema does not support multiple network endpoints to the same service instance

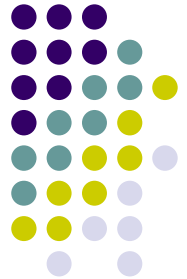
  Or

- The current schema does not support the representation of a single resource with multiple/alternative service instances to access it
  - E.g.: a single cluster with two head-nodes/gatekeepers appears as two different clusters?

Multiple Endpoints to the Same Service

- Idea: the core concept to be modeled is the resource, not the service
- The resource has a URN (Uniform Resource Name)
- The resource may have one or more services
- A default service/endpoint should be identifiable
  - If I have multiple service instances to the same resource, how do I choose among them?
  - Can they be accessible from different VO/groups?

JSDL and GLUE Schema

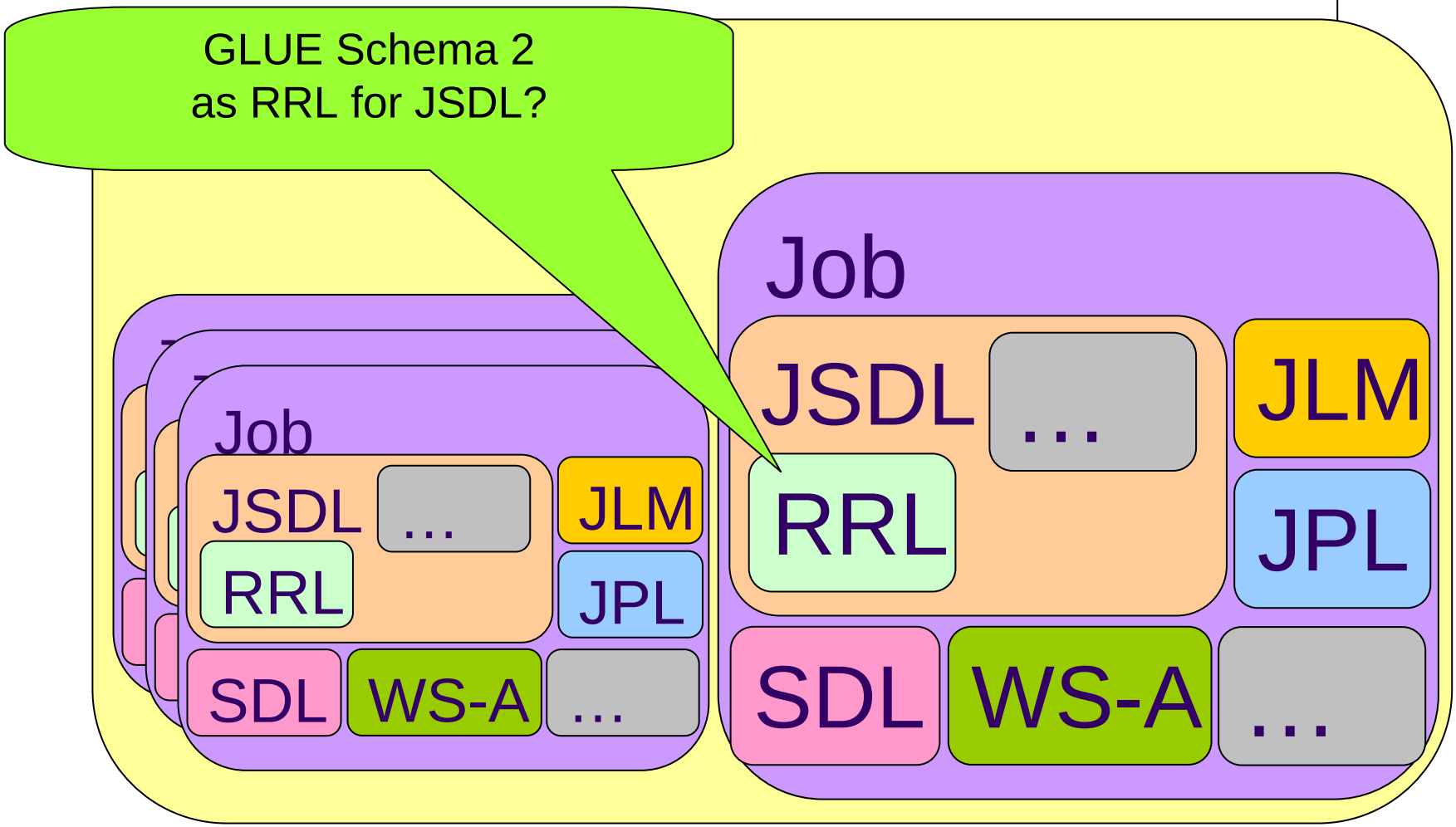


- JSDL: an OGF standard defining an extensible language to describe job submission
- Job submission implies expressing requiremetns on resources

gLite.JDL : GLUE Schema = OGF.JSDL : RRL

To my knowledge, no external RRL for JSDL exists today;
an internal set of resource-related attributes exist

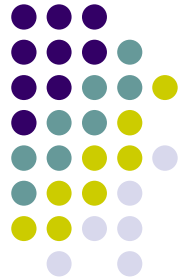
JSDL and GLUE Schema



RRL - Resource Requirements Language SDL - Scheduling Description Language WS-A - WS-Agreement

JLM - Job Lifetime Management JPL - Job Policy Language

JSDL and GLUE Schema



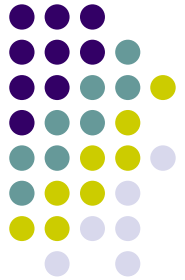
- Aspects to be considered:
  - GLUE Schema is a description from the resource viewpoint
    - E.g.: Glue.CE.Policy.MaxCPUTime
  - In gLite.JDL, requirements are expressed as predicate over resource properties
    - E.g.: GlueCEPolicyMaxCPUTime>600
- In JSDL, the requirements are stated not as predicate, but as attributes related to acceptable thresholds:
  - <jsdl:TotalCPUTime>
  - <jsdl:UpperBoundedRange>600.0</jsdl:UpperBoundedRange>
  - </jsdl:TotalCPUTime>

IMPORTANT ISSUE TO BE CONSIDERED

GLUE Schema and CIM /1

- GLUE Schema
  - tailored approach to the modeling of Grid resources for brokering, access and monitoring
  - Principles:aggregation, generalization and denormalization
  - Description in UML, mapping into different data models
  - HEP-Grid community standard (expanding adoption)
- CIM
  - modeling of IT resources for their management
  - Principles: fine-grained description and normalization
  - description in UML, mapping into concrete data models (mainly XML)
  - DMTF standard
  - Part of WBEM to provide infrastructure for management comprising interface, protocol and query language to interact with a managed resource

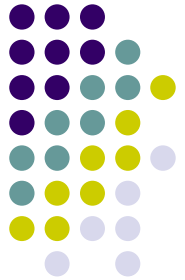
A Dream Solution?



- GLUE Schema 2
  1. Start from 1.3
     1. Remove deprecated attributes
     2. Clean all the bad design
  2. Address new requirements from new projects
  3. Cast it into the CIM meta-model
  4. Derive a RRL for JSDL 1.x
  5. Derive a schema for the various information services
  6. Write a suite of test queries for the different mappings

Is this feasable?

CONCLUSION



- We have strong experience on production Grids
- The integration to OGF standards is an important goal, nevertheless it is not going to be easy
- What is the best way to approach this major evolution?