MANCHESTER 1824

The University of Manchester

Brokering in UNICORE

John Brooke and Donal Fellows
University of Manchester

john.brooke@manchester.ac.uk

donal.k.fellows@manchester.ac.uk

Unicore Summit
11-12 October, 2005
Sophia Antipolis, France.

Grids as Virtual Organizations

•Used in paper Anatomy of the Grid (Foster, Kesselman, Tuecke)

•“ … Grid concept is coordinated resource sharing in dynamic, multi-institutional virtual organizations …”

•The link to the Power Grid concept is that the Power Grid is resource sharing by producers to provide a unified service to consumers.

•A large unresolved question is how do Virtual Organisations federate across security boundaries (e.g. firewalls) and organisational boundaries (resource allocation).

The analogy with a power grid

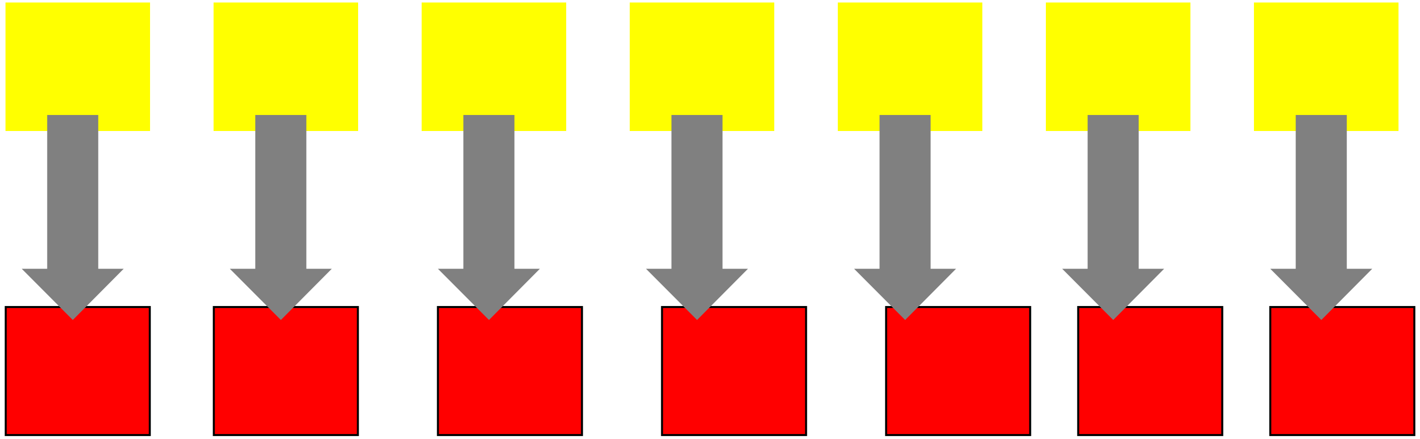
- The power grid delivers electrical power in the form of a wave (A/C wave)
- The form of the wave can change over the Grid but there is a universal (scalar) measure of power, Power = voltage x current.
- This universal measure facilitates the underlying economy of the power grid. Since it is indifferent to the way the power is produced (gas, coal, hydro etc…) different production centres can all switch into the same Grid.
- To define the abstractions necessary for a Computational Grid we MUST understand what we mean by computational resource.

The University of Manchester

MANCHESTER 1824

An abstract space for job-costing

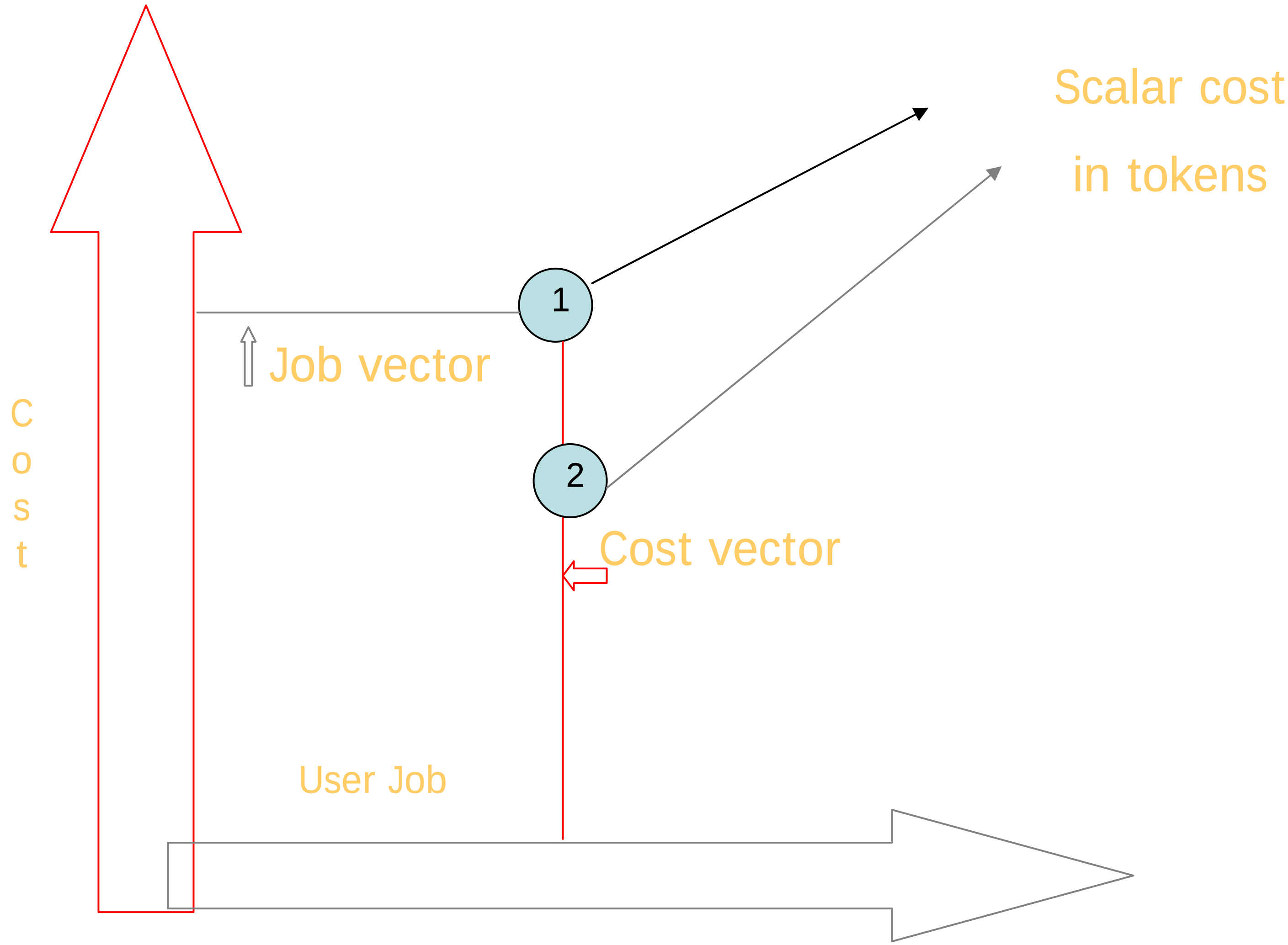
- Define a job as a vector of computational resources
  - `(r1,r2,…,rn)`
- A Grid resource advertises a cost function for each resource
  - `(c1,c2,…,cn)`
- Cost function takes vector argument to produce job cost
  - `(r1*c1 + r2*c2 + … + rn*cn)`



A Dual Job-Space

- Thus we have a space of “requests” defined as a vector space of the computational needs of users over a Grid. For many jobs most of the entries in the vector will be null.
- We have another space of “services” who can produce “cost vectors” for costing for the user jobs (providing they can accommodate them).
- This is an example of a dual vector space.
- A strictly defined dual space is probably too rigid but can provide a basis for simulations.
- The abstract job requirements will need to be agreed. It may be a task for a broker to translate a job specification to a “user job” for a given Grid node.

4 - Dual Space


Scalar cost
in tokens
1
Job vector
2
C
o
s
t
Cost vector
User Job

Computational resource

- Computational jobs ask questions about the internal structure of the provider of computational power in a manner that an electrically powered device does not.
- For example, do we require specific compilers, libraries, disk resource, visualization servers?
- What if it goes wrong, do we get support? If we transfer data and methods of analysis over the Internet is it secure?
- A resource broker for high performance computation is a different order of complexity to a broker for an electricity supplier.

Emergent behaviour

- Given this complexity, self-sustaining global Grids are likely to emerge rather than be planned.
- Planned Grids can be important for specific tasks, the LCG is an example. They are not required to be self-sustaining and questions of accounting and resource transfer are not of central interest.
- Self-sustaining (in a financial sense) Grids must have ways of accounting for and costing resource.
- This allows for trading mechanisms between suppliers of computational resource analogous to trading in energy and power Grids

Resource Requestor and Provider Spaces

- Resource requestor space (RR), in terms of what the user wants: e.g. Relocatable Weather Model, 10^6 points, 24 hours, full topography.
- Resource Provider space (RP), 128 processors, Origin 3000 architecture, 40 Gigabytes Memory, 1000 Gigabytes disk space, 100 Mb/s connection.
- We may even forward on requests from one resource provider to another, recasting of O3000 job in terms of IA64 cluster, gives different resource set.
- Linkage and staging of different stages of workflow require environmental support, a hosting environment.

What is Computational Brokering?

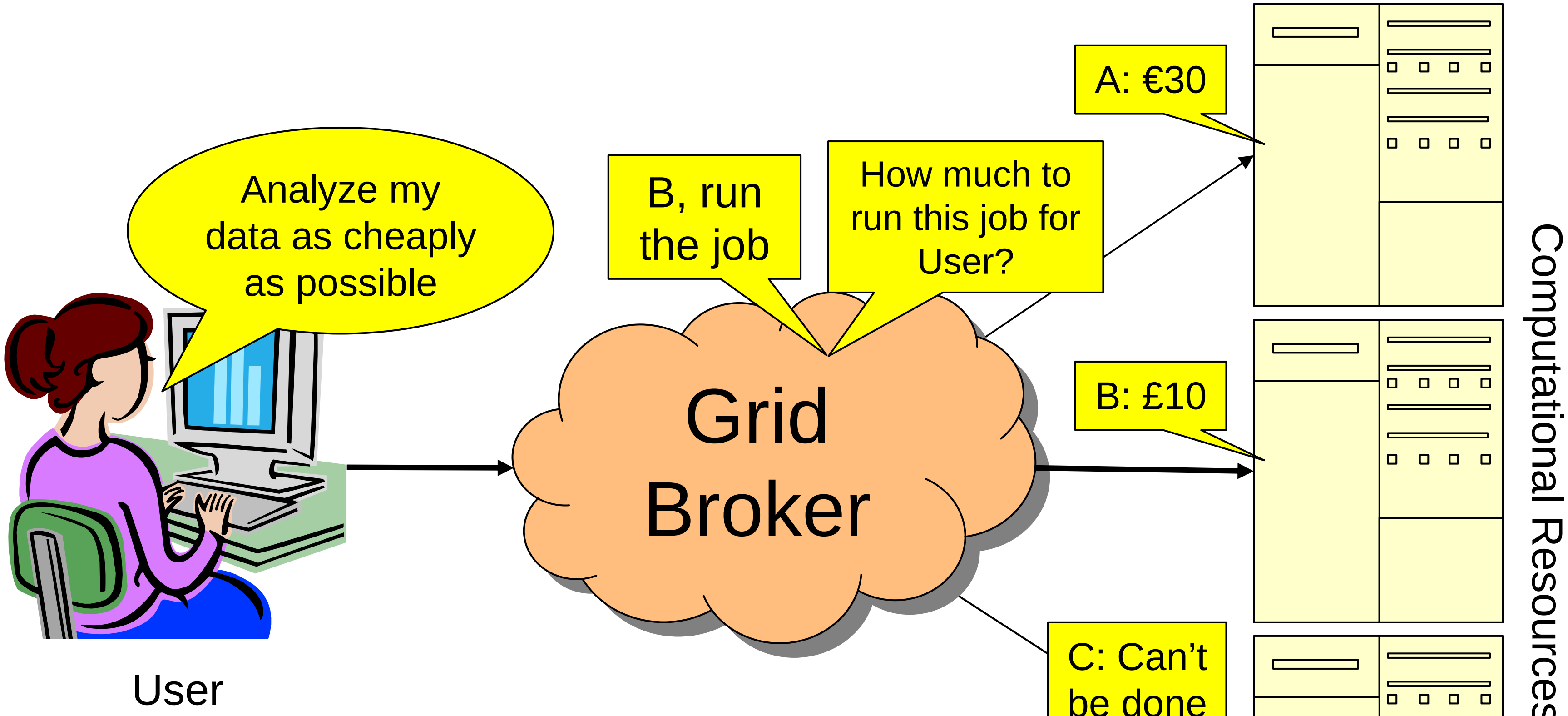
- Many Definitions!
  - Selection of where to run a job
  - Selection of what jobs to run on a resource
  - Selection of when to run a job
  - Discovery of how to run a job
  - Generalized user representative agent

- We Focus on “Where to Run a Job”

Why Broker over a Grid?

- Discover Resources
  - Users *should not* know all places to run jobs
- Discover Costs
  - A typical pricing model is complex
  - Sites may wish to conceal information from competitors
- Discover Queuing and Running Times
  - Some systems are very fast, but use batch queues
  - Some systems are slower due to competition, but offer immediate execution
- Improve Resource Utilization
- Account for Partial Resource Availability

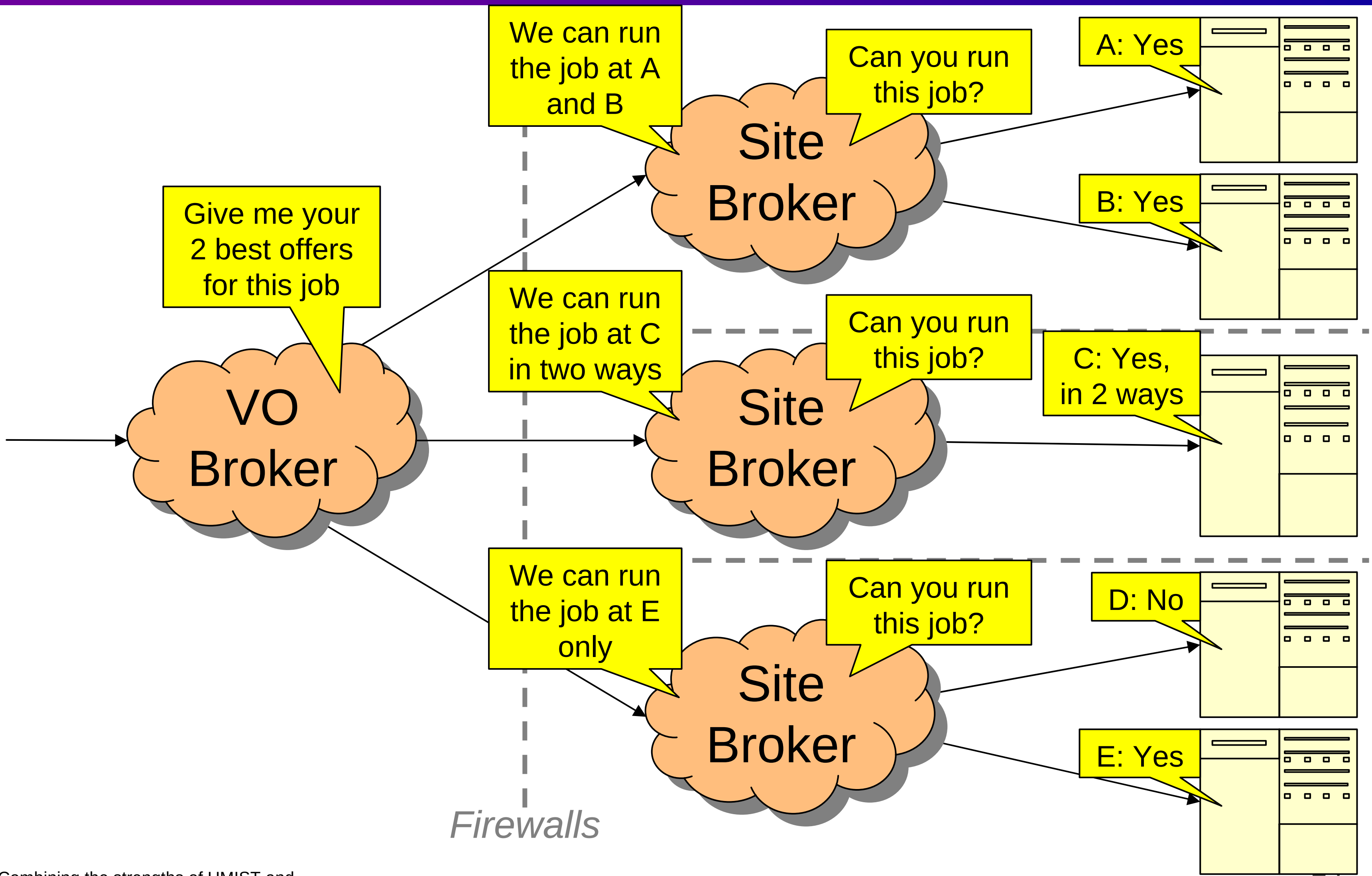
Grid Brokering



Why *Distributed* Brokering?

- Sites Know the State of their Resources Best
- Sites Can Conceal their Resource Configuration
- Different VOs Need Different Selection Algorithms
  - Preferred site sets will vary
  - Different applications have different performance characteristics
- Divide-and-Conquer

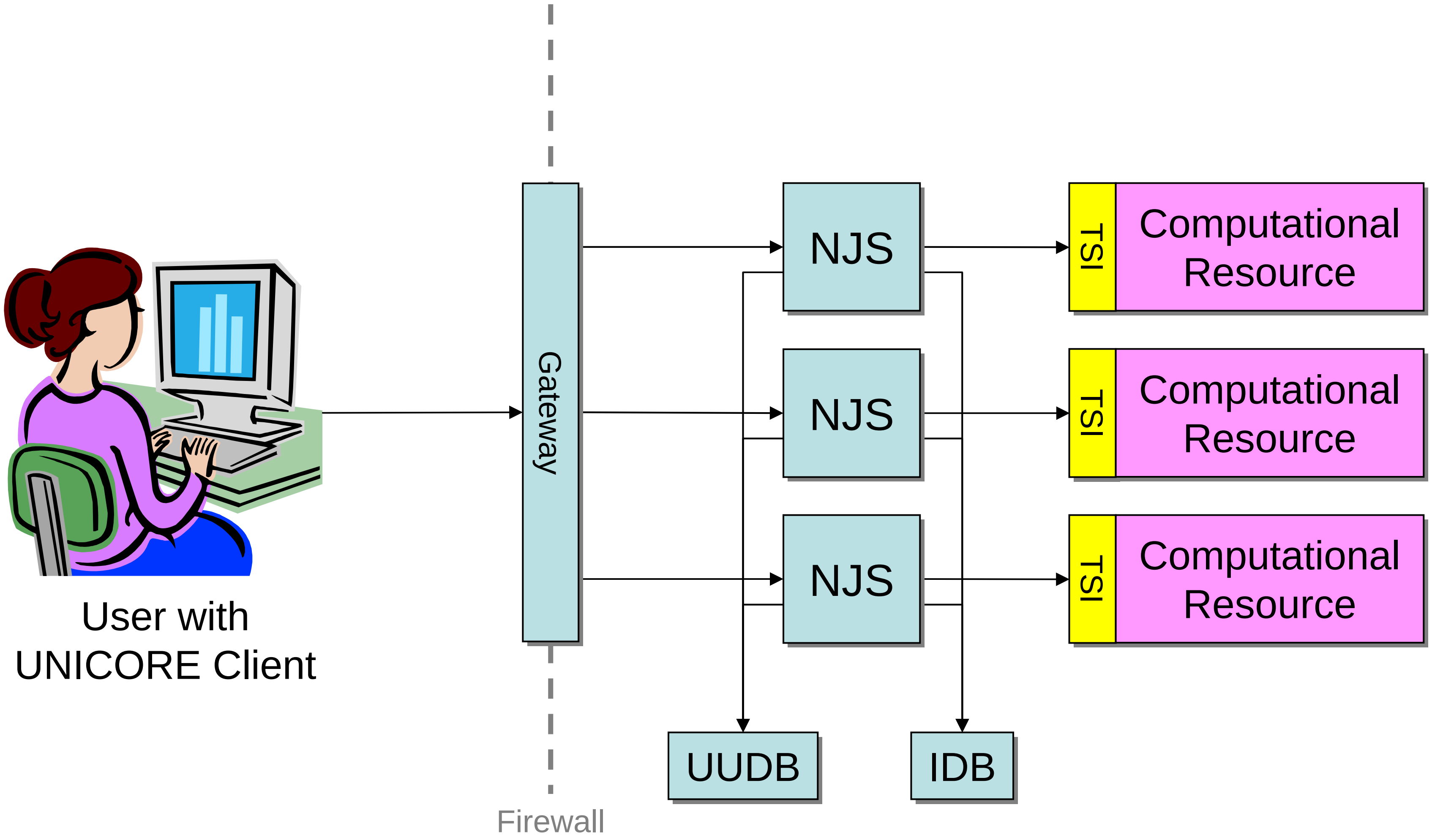
Distributed Brokering



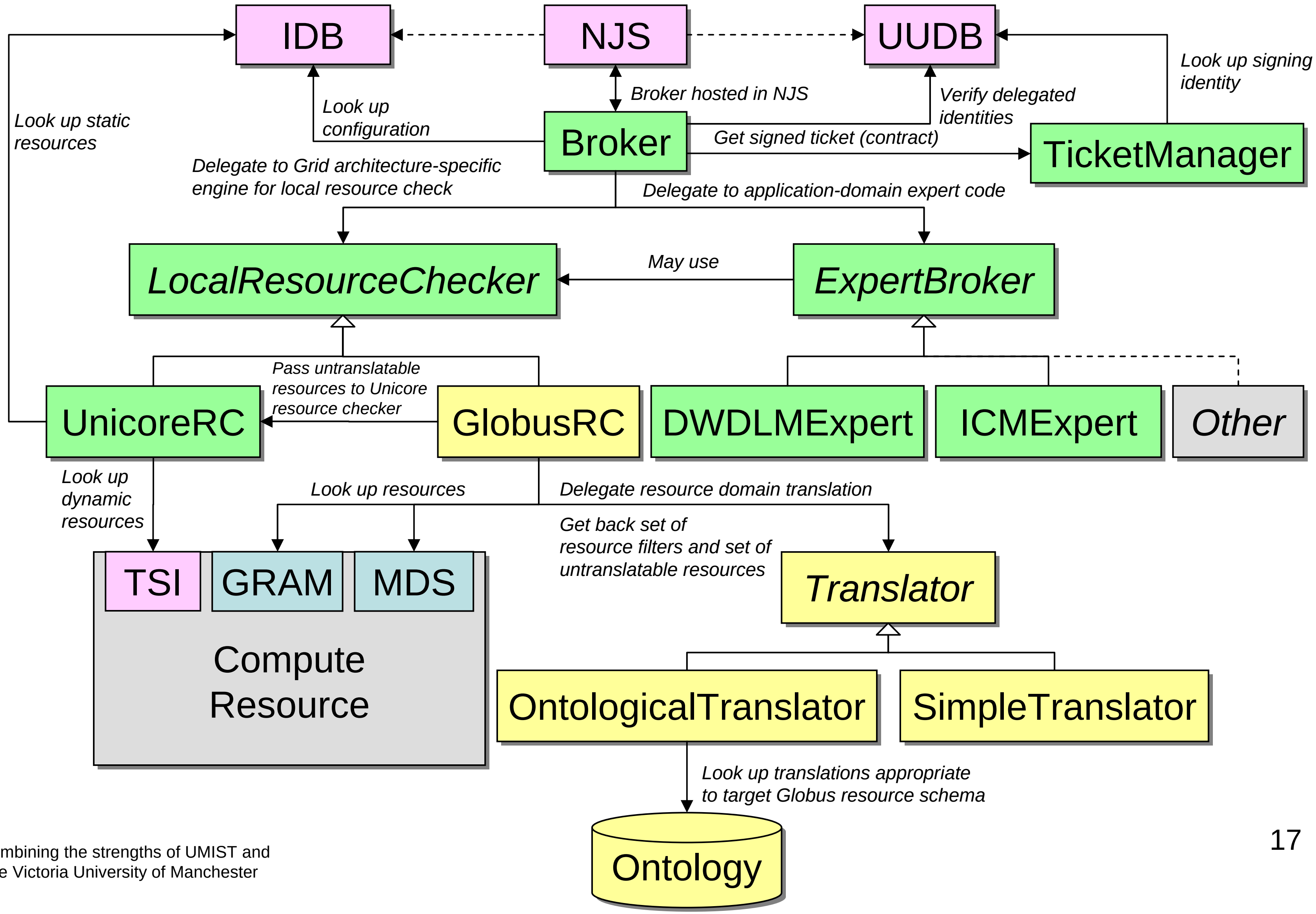
The UNICORE Broker

- Flexible
  - Can issue multiple alternative offers per resource
  - Can issue offers for multiple resources
- Distributed
  - Delegate brokering requests to other brokers
- Powerful
  - Workflow brokering
  - Supports application-specific brokering
- Interoperable
  - Supports transparent brokering across both UNICORE and Globus grids

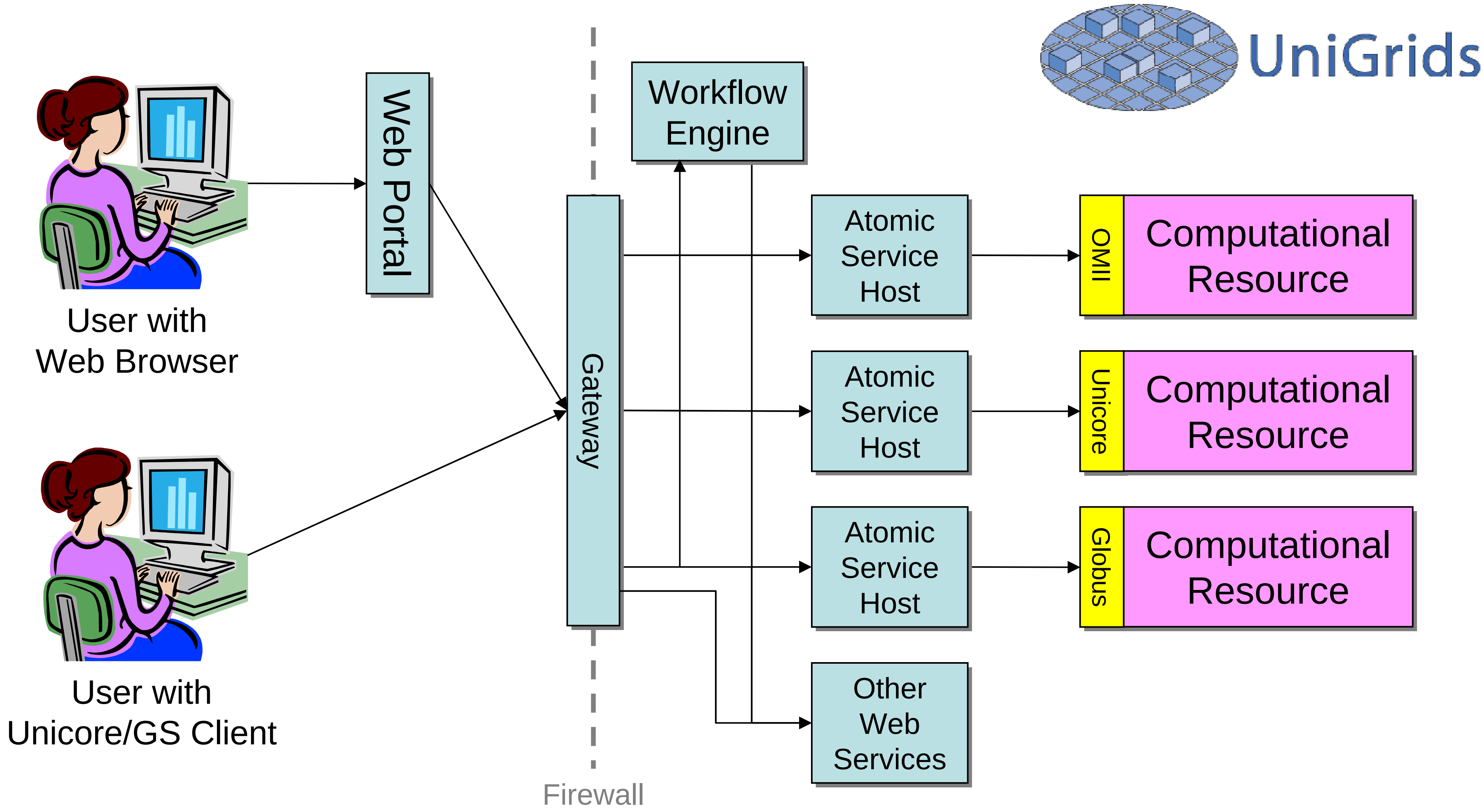
UNICORE Architecture



UNICORE Broker Internal Architecture



Unicore/GS Architecture



Combining the strengths of UMIST and
The Victoria University of Manchester

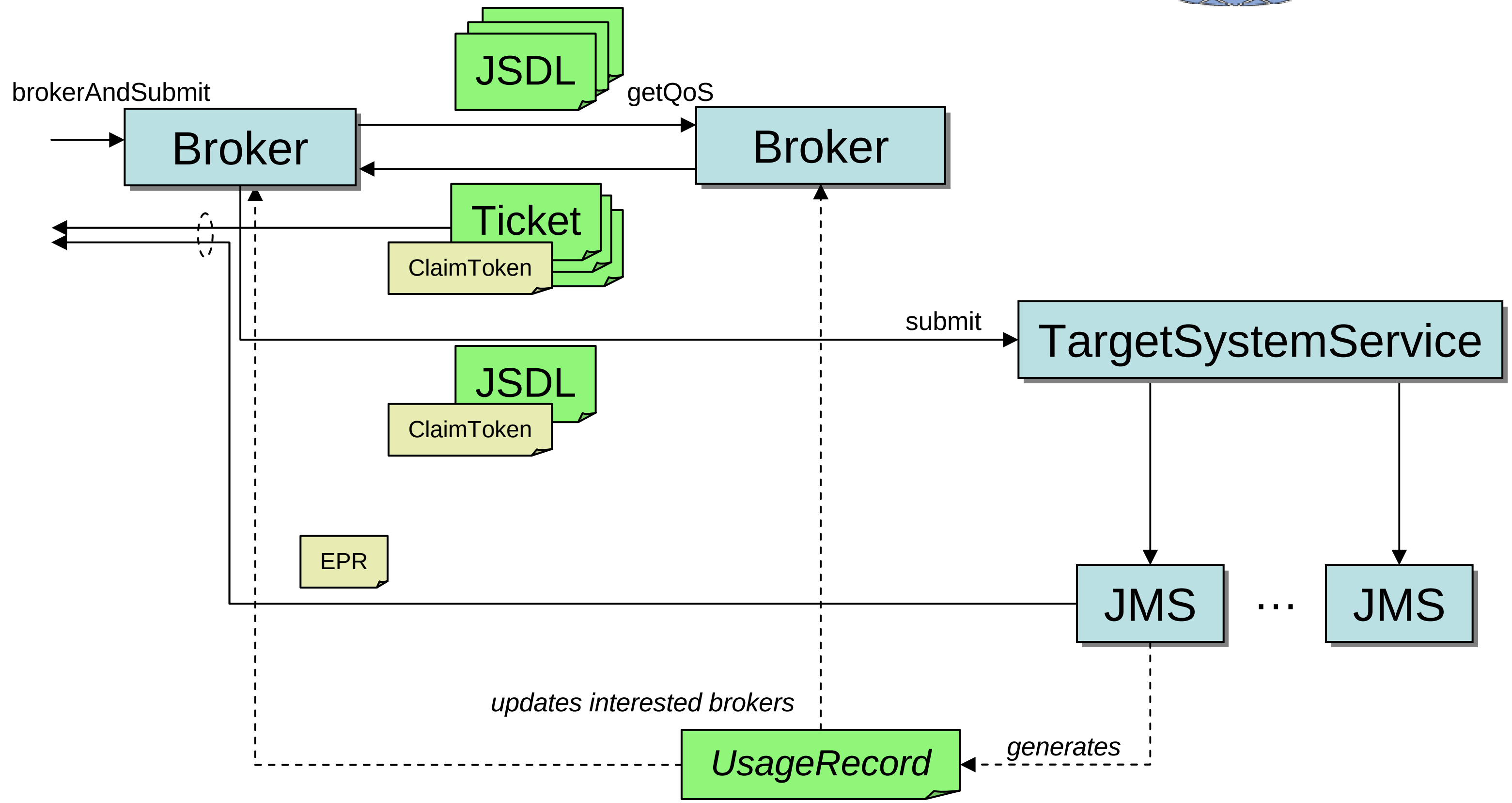
The University of Manchester

MANCHESTER 1824

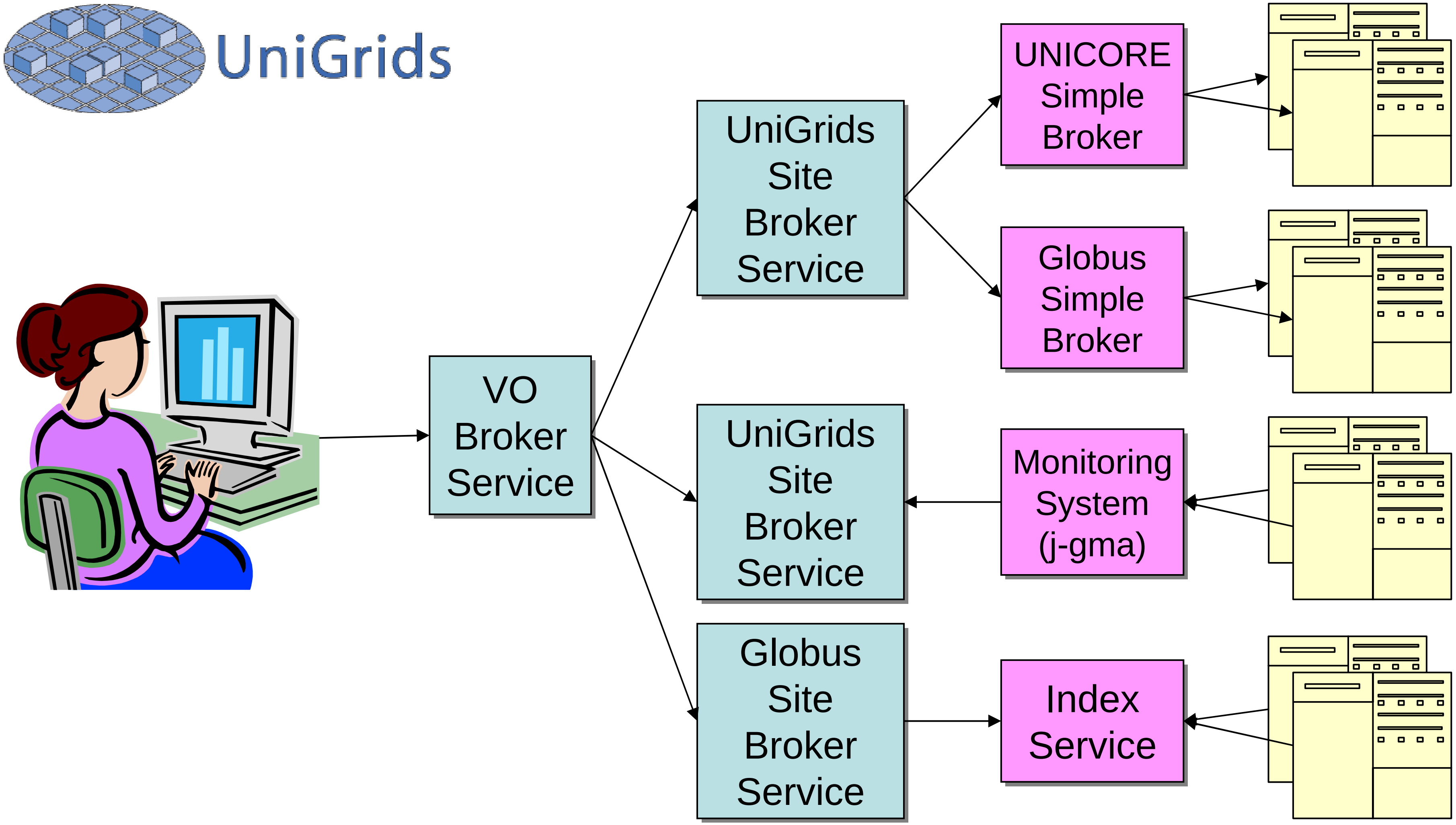
UNICORE

UniGrids Broker Service Architecture

UniGrids



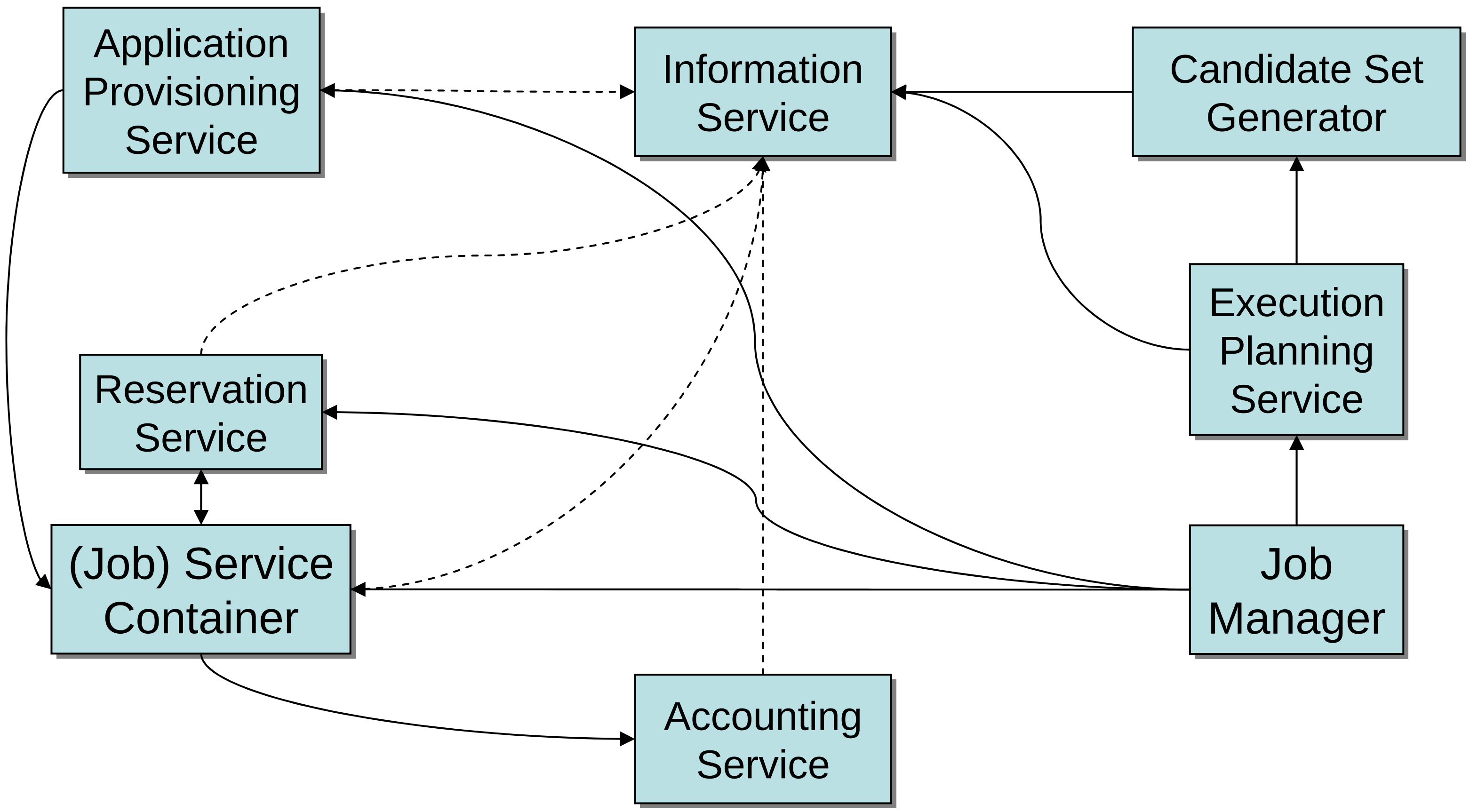
UniGrids Grid Brokering Architecture



OGSA

- **O**pen **G**rid **S**oftware **A**rchitecture
  - Open standard under development
  - Both business and academia
- Fundamental Architecture of Future Grids
  - Based on web-services
    - Service-oriented
  - Covers many areas of basic grid systems
- Major Aspect is Execution Management

OGSA Execution Management

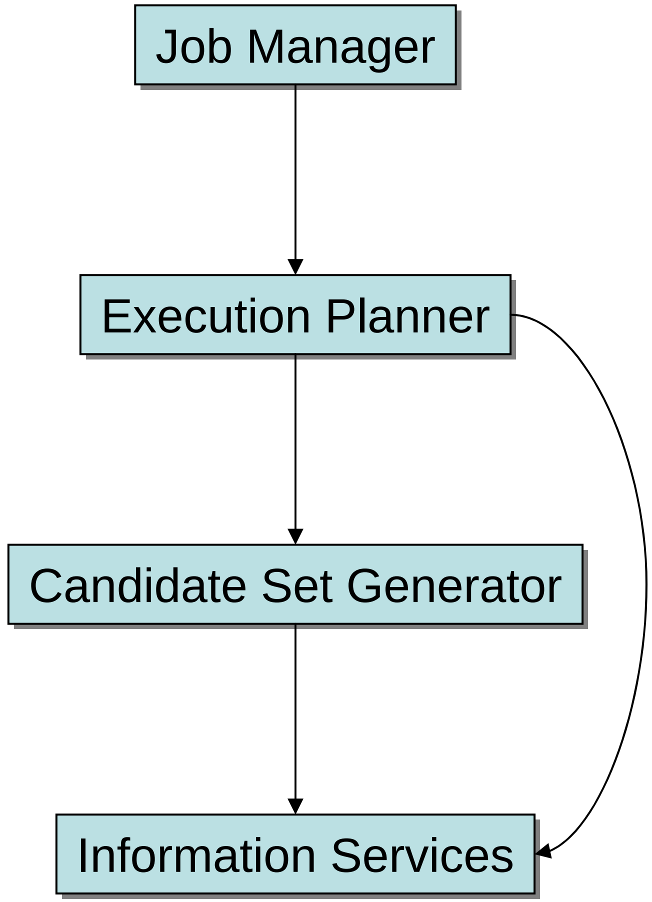


OGSA Resource Selection

- Part of OGSA Execution Management
- Focuses on Choosing Which Service to Use
  - Brokering
  - Super-scheduling
- Partitioned Selection System
  - CSG suggests where to place atomic tasks
  - EPS creates overall plans for workflows
    - Based on cost, time to execute, etc.

OGSA Resource Selection Architecture

- JM asks EPS for ways to run a workflow
- EPS asks CSG for ways to instantiate atomic tasks
- CSG returns set of candidate executions
  - Where, how, etc.
- EPS picks good combinations for overall workflow
  - Task combination is non-trivial
  - Uses info-services to discover additional costs of combination
- JM uses the resulting plan(s)
- *Very* similar to UNICORE Broker architecture



Summary

- UNICORE Broker is Flexible Agent Supporting Rich Distributed Grid Brokering
  - Workflows
  - Application-specific brokering
  - Works with multiple low-level grids
  - Has been deployed in grid across multiple organizations
- UniGrids Broker is Next Evolutionary Phase
  - Driving OGSA standards
  - Leveraging low-level grid information systems

MANCHESTER 1824

The University of Manchester

UNICORE

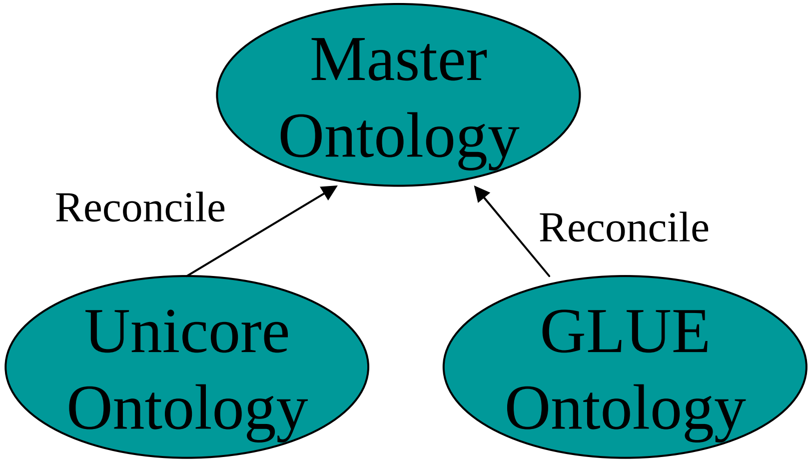
Grid Resource Description Problem

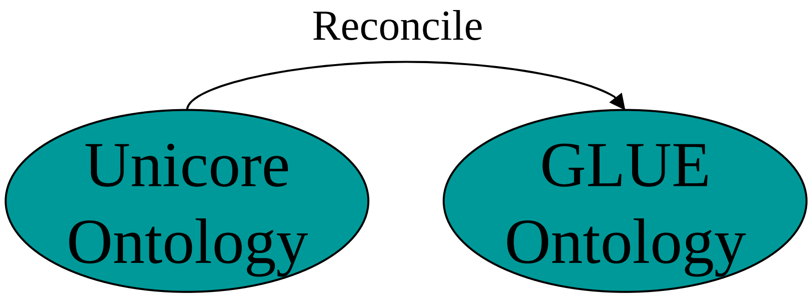
- Two Independent Grid Systems
  - Unicore (http://www.unicore.org/)
  - Globus (http://www.globus.org/)
- Both Need to Describe Systems that run Compute Jobs
- Very Different Description Languages
  - Unicore's Resource model, part of the AJO Framework
  - Globus's GLUE Schema (DataTAG, iVGDL) for GT2 and GT3
- For interoperability, we want to take a Unicore job and run it on Globus resources
- Therefore, we need to translate the Job's Resource Requirements between the two Systems

Two Types of Integration

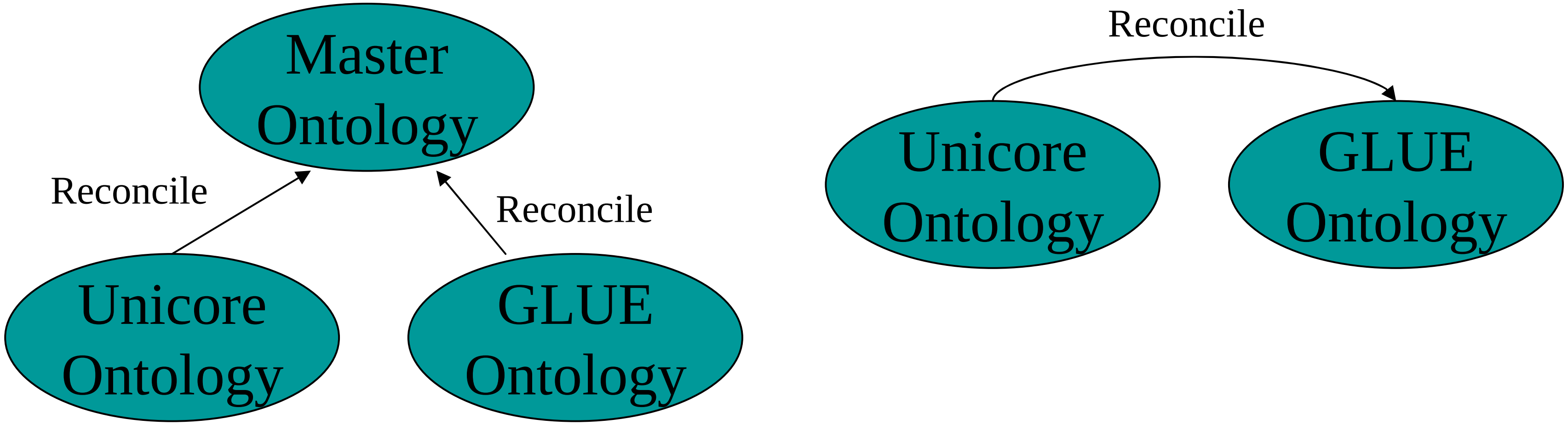
- Right here, Right now:
  - Integrate existing features such as the way Unicore and Globus currently describe hardware resources
  - Best done by evolution, preserving much of the character of the legacy system components
- The future:
  - Integrating future features such as the way Unicore and Globus will describe Software Resources
  - Best done by revolution, introducing a new system, reached by consensus between the two teams of architects

Methodology for translation





Methodology for translation

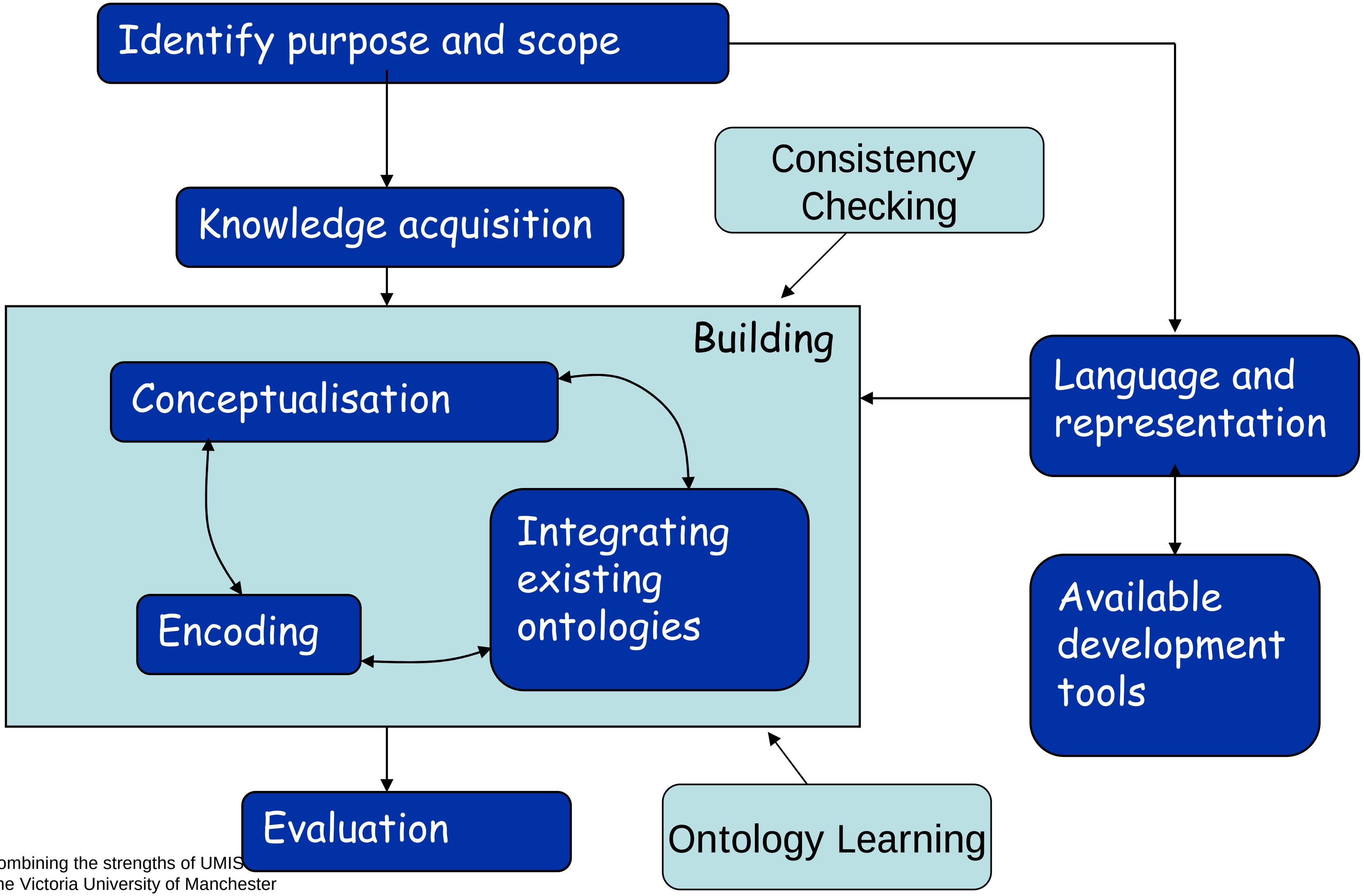


- Develop an ontology for the Unicore resource terminology
- Develop an ontology for the Globus resource terminology
- Map concepts in the Unicore ontology to concepts in the Globus ontology
- We assume a consensus between the concepts in Unicore and GLUE

Methodology for translation service

- Address Data Transformation Issues for Translating Attributes
- Find a technology that has these characteristics:
  - can model the two ontologies
  - has support for linking abstract concepts to code fragments
  - easily allows someone to update mappings
  - is appropriate for a video conferencing setting
  - writes modelling information to a file format that can be used by other applications
- Use the data files created by the application to run the translator service.

An Ontology Building Life-cycle



MANCHESTER 1824

The University of Manchester

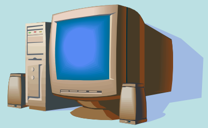
UNICORE

Ontology evolution: From Architects to Software

Service User

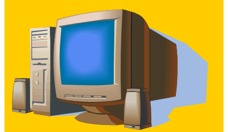
Unicore Architect

Unicore



Service provider

GLUE Architect



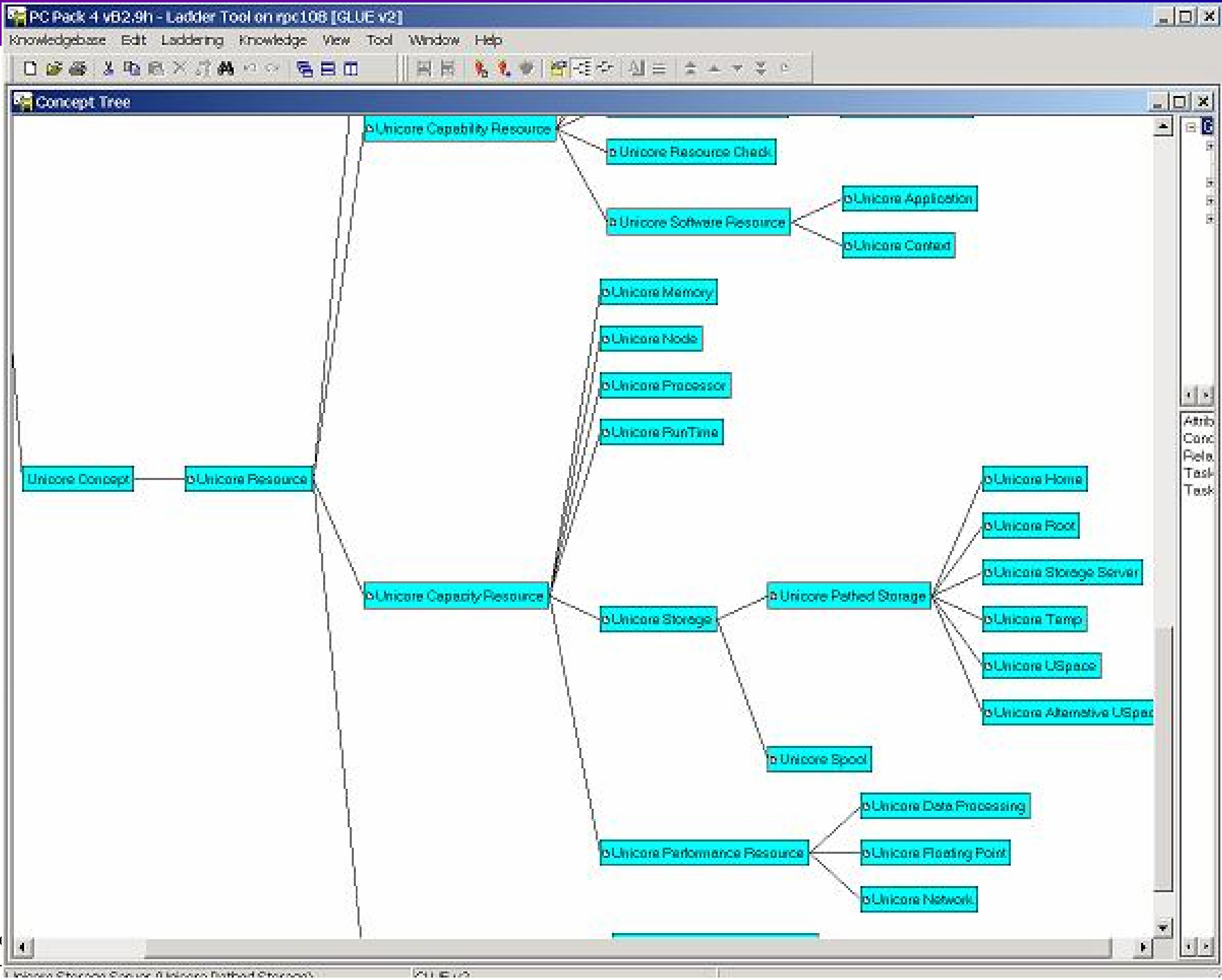
Globus

Brief Descriptions
Infer a lot from context

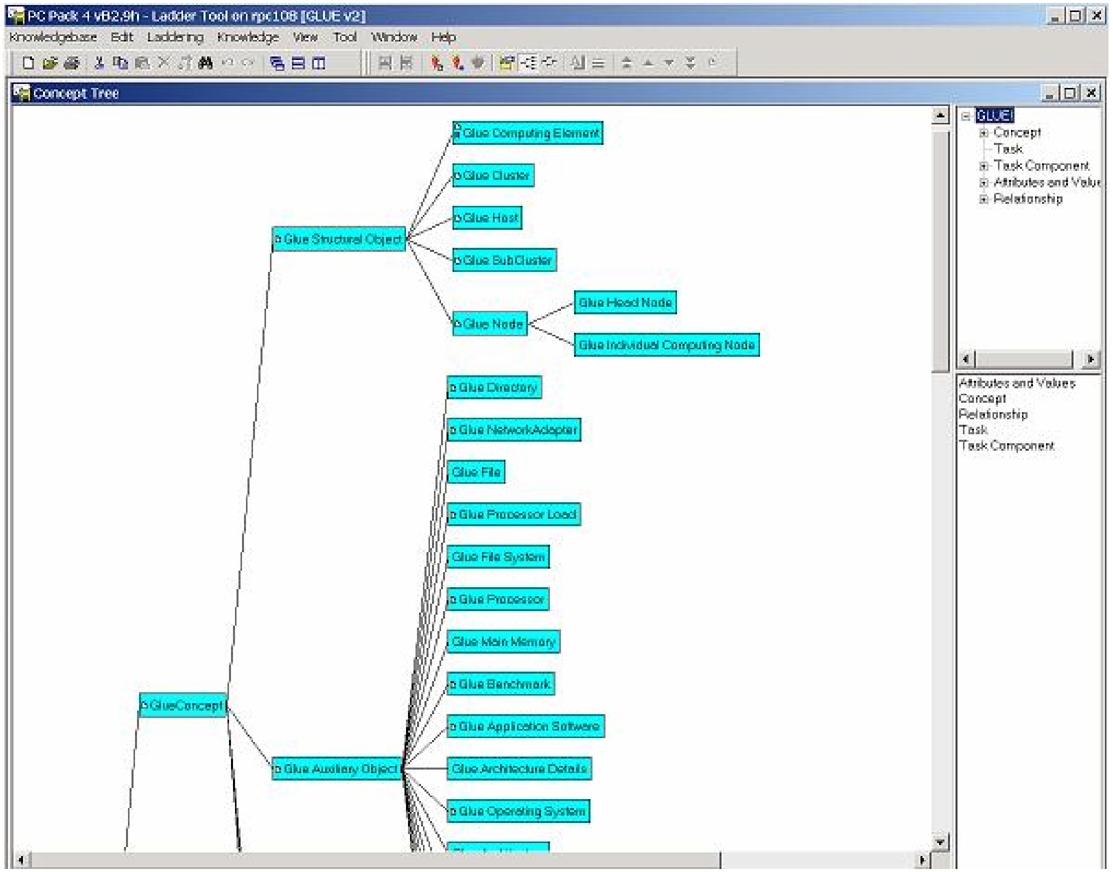
Syntactic descriptions
Interface descriptions
Invocation descriptions

Semantic mining

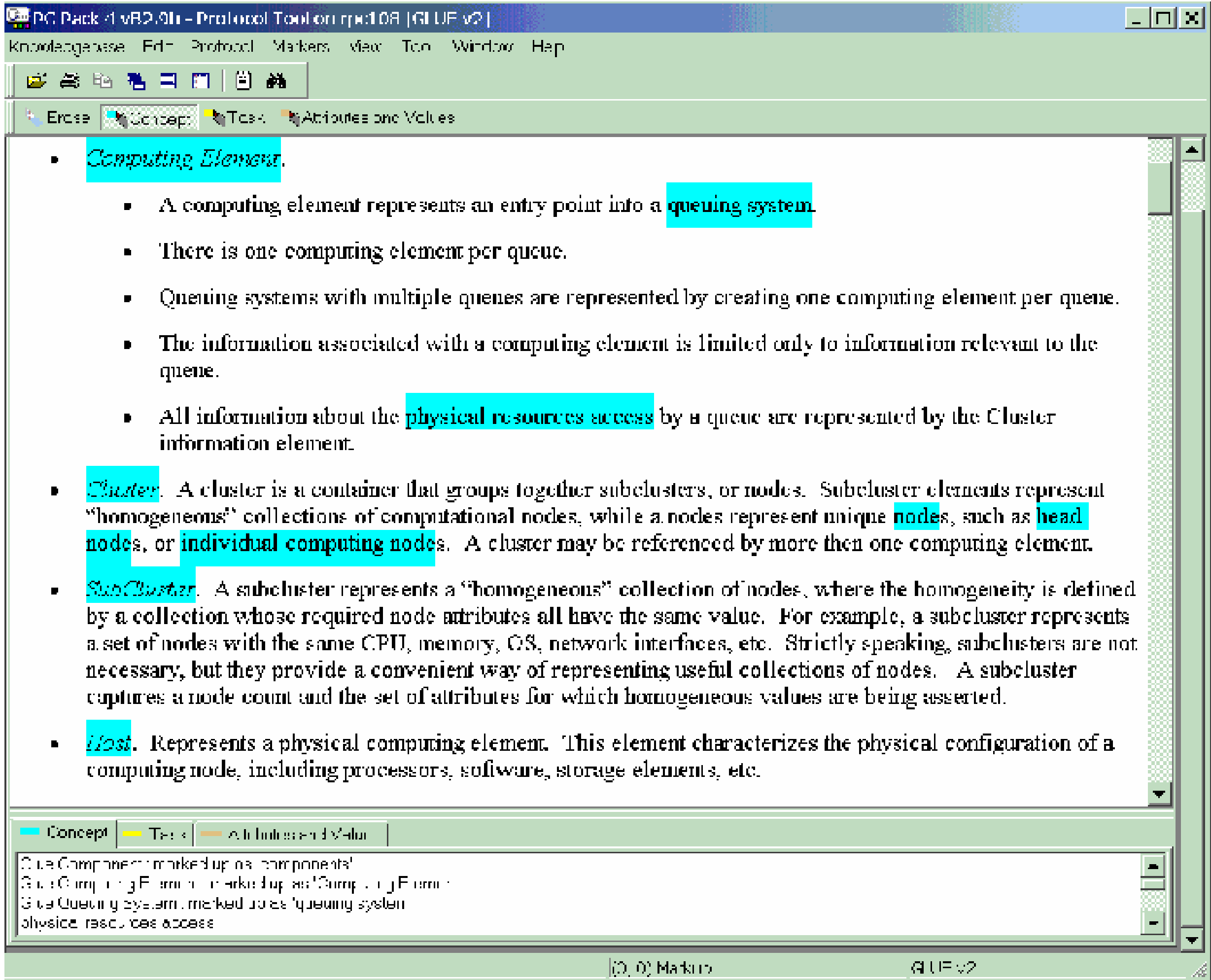
Unicore: Modelling Resources



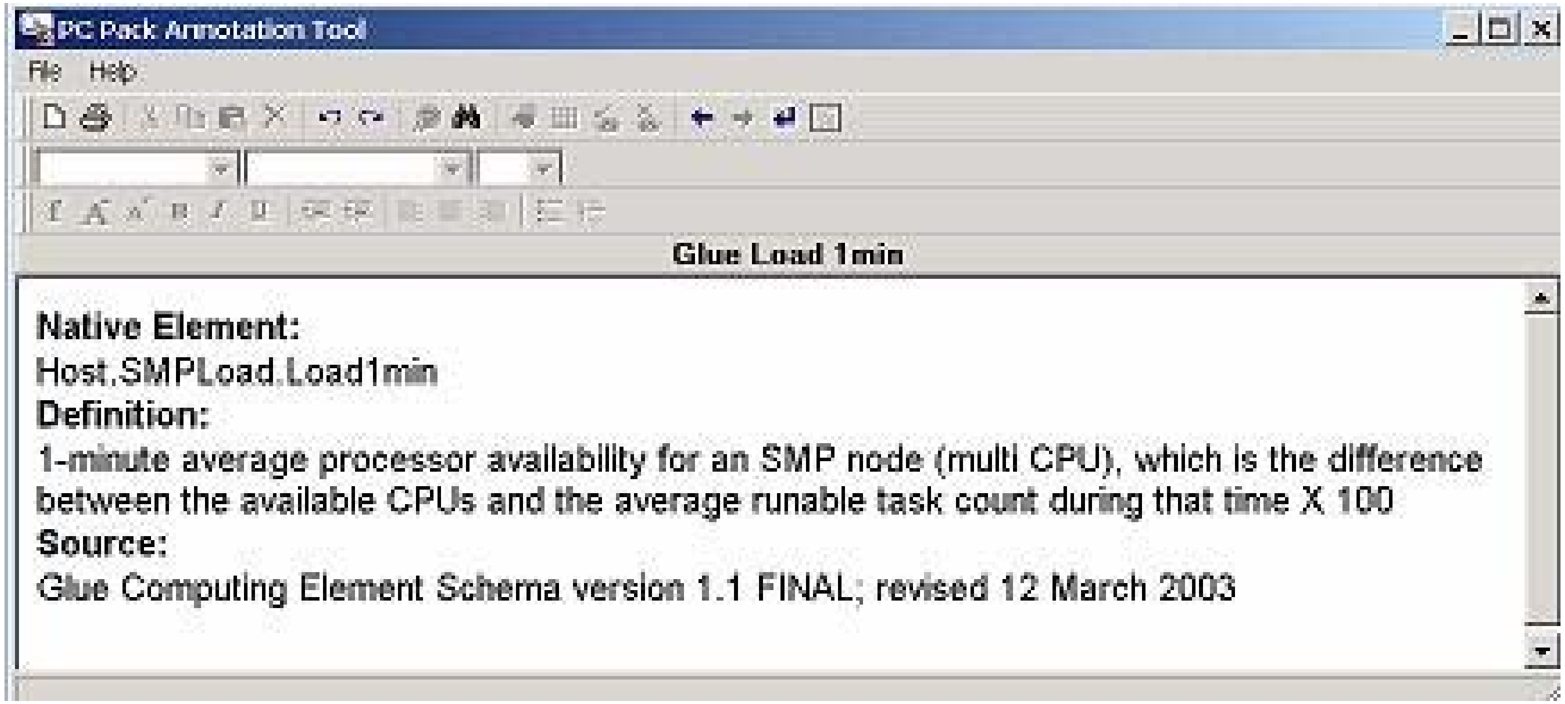
GLUE: Modelling resources



GLUE: Marking up transcripts



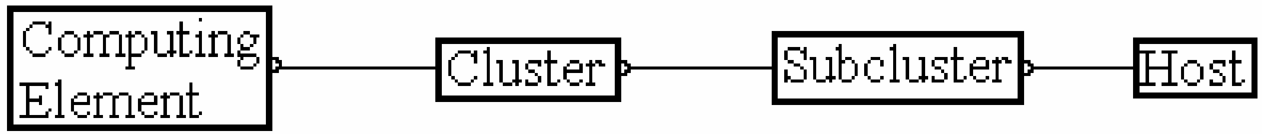
GLUE: Provenance Information



GLUE: Container Classes

- GLUE has container classes that include “Computing Element”, “Cluster”, “Subcluster” and “Host”. From the heading “Representing Information”, the GLUE document indicates:

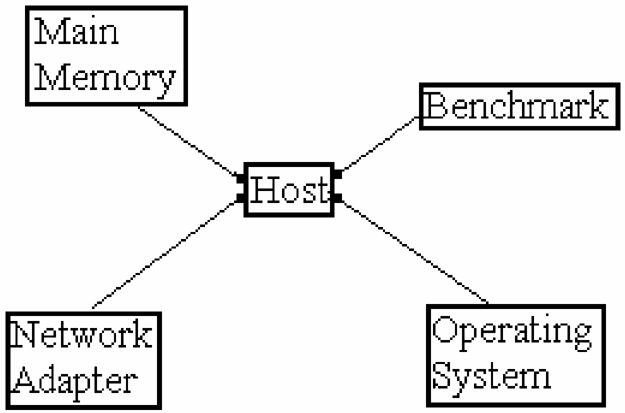
  “…hosts are composed into sub-clusters, sub-clusters are grouped into clusters, and then computing elements refer to one or more clusters.”



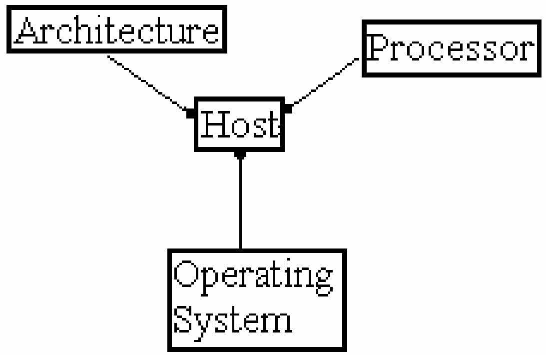
- These container objects may hold any number optional auxiliary classes that actually describe the GRID features.

GLUE: Auxiliary Classes

- The documentation provides few details about the nature of a Host other than that it is a “physical computing element”. Much of the meaning for Host has to be derived from what it might contain. Consider the following two valid definitions:

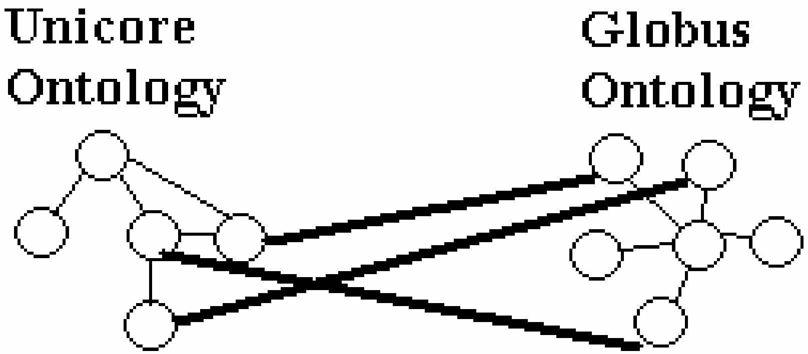


- A Host is a physical computing element characterized by Main Memory, a Benchmark, a Network Adapter and an Operating System



- A Host is a physical computing element characterized by an Architecture, a Processor and an Operating System.

Map concepts between ontologies

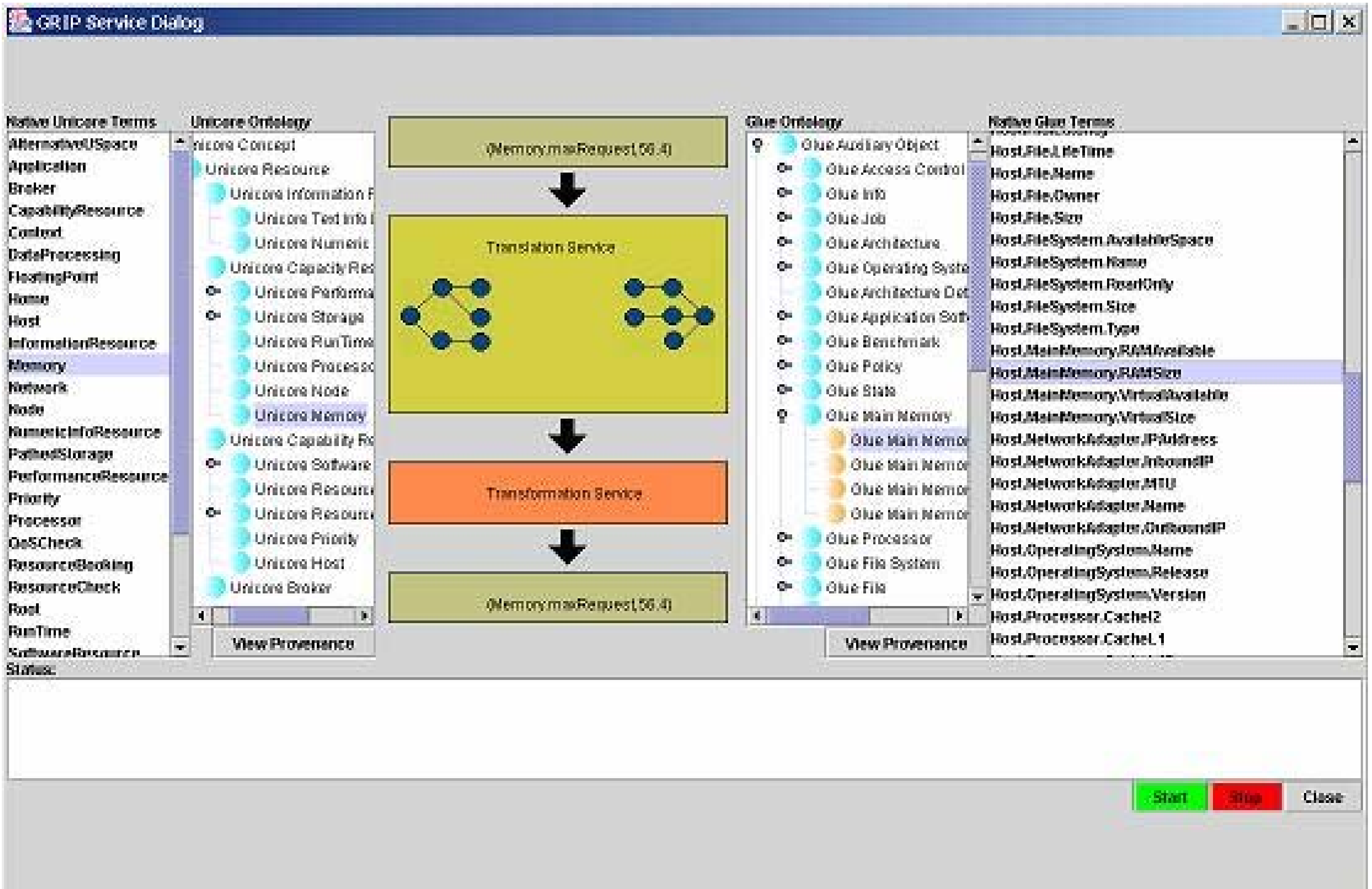


- Unicore and GLUE have different philosophies for describing resources :-(
- In Unicore, the resources are described in terms of resource requests
- In GLUE, resources are described in terms of the availability of resources.

Compatible Concepts

Unicore Ontology	GLUE Ontology
Network Performance	Glue SI00 Benchmark
Network Performance	Glue SF00 Benchmark
Floating Point Performance	Glue SI00 Benchmark
Floating Point Performance	Glue SF00 Benchmark
Data Processing Performance	Glue S100 Benchmark
Data Processing Performance	Glue SF00 Benchmark
Maximum Memory Capacity Request	Host Virtual Main Memory Available
Maximum Memory Capacity Request	Subcluster Virtual Main Memory Available
Minimum Memory Capacity Request	Host RAM Main Memory Available
Minimum Memory Capacity Request	Subcluster RAM Main Memory Available
Priority Value	Priority

Translation Service Prototype



The End

Questions?

donal.k.fellows@manchester.ac.uk