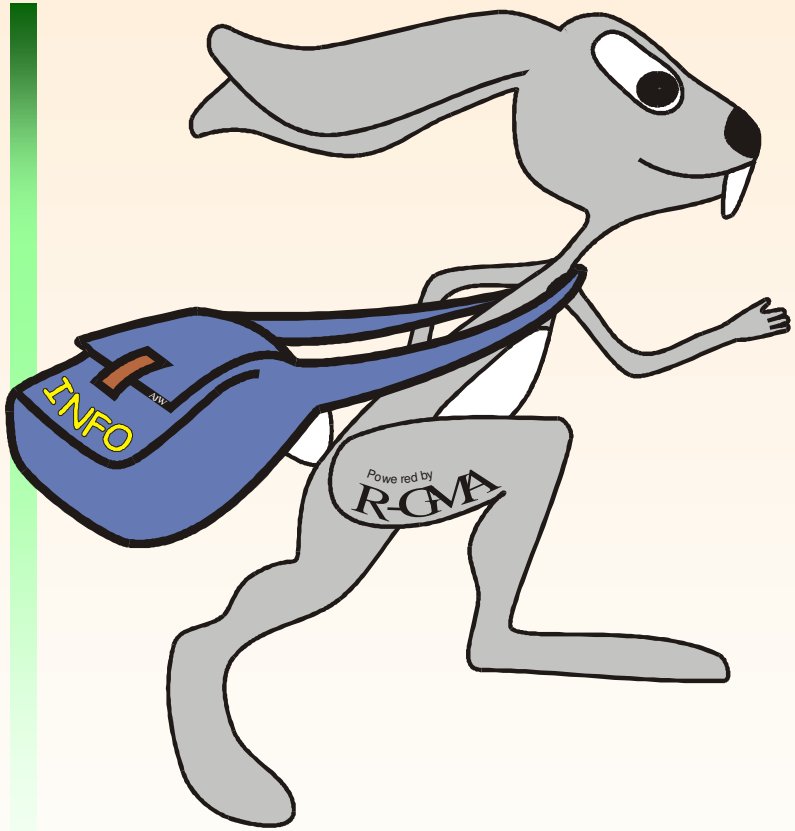


Implementing R-GMA grid services in GT3

Abdeslem Djaoui &
WP3 Grid Services Task Force

7th EU Datagrid meeting 26/09/2003

a.djaoui@rl.ac.uk

Outline

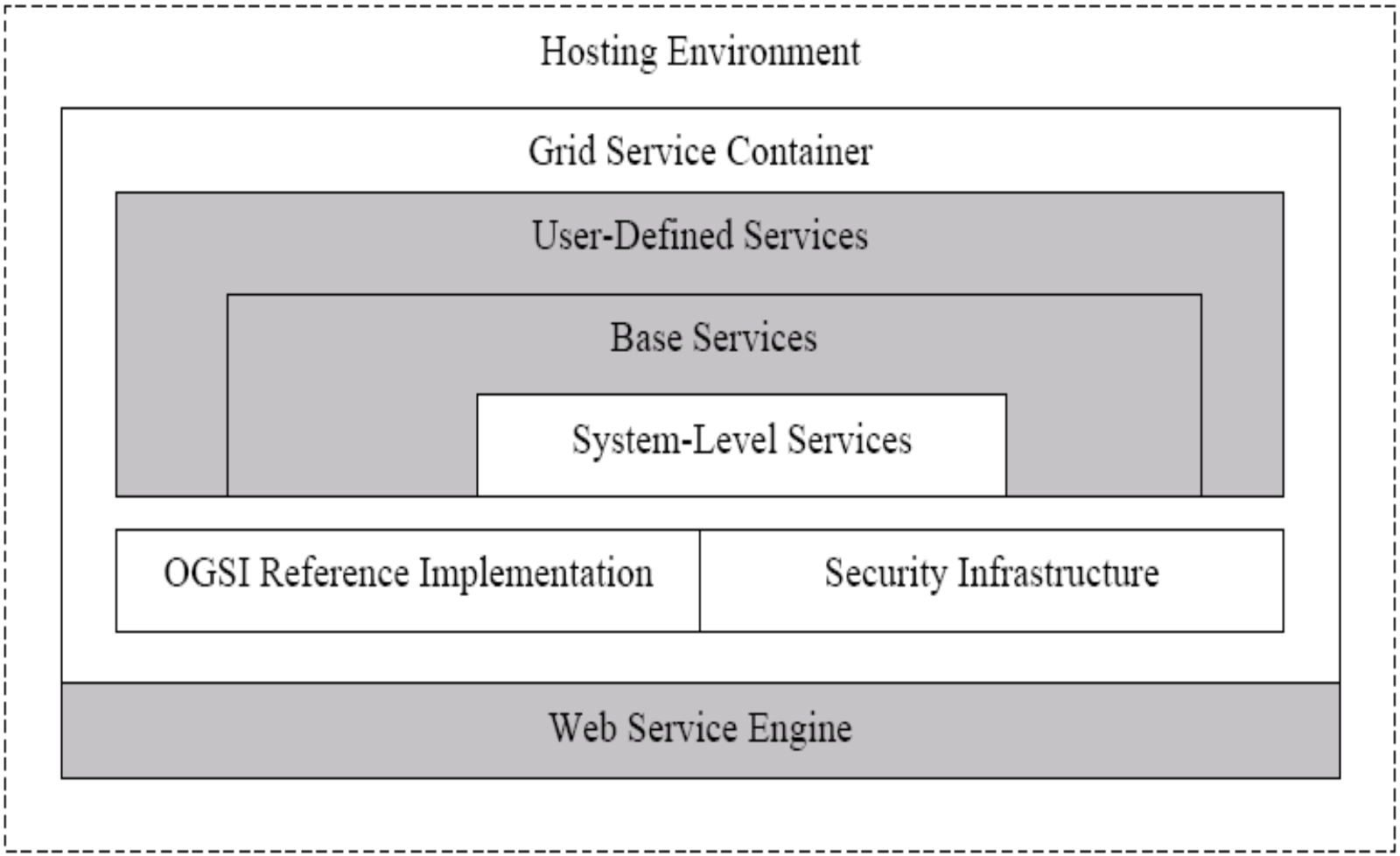
- GT3
- R-GMA schema grid service

- Next presentation: Demo by James

What is GT3

- Open source reference implementation of OGSI
  - GT3 core
- Plus several OGSI-compliant services corresponding to familiar GT2 services
  - Grid security: GSI
  - Remote job submission and control: GRAM
  - High peformance secure data transfer: GridFTP
  - System and service information: MDS
- Is also a framework for the creation of new OGSI-compliant services

GT3 core architecture – white boxes



GT3 OGSI reference implementation

- Set of primitives implementing the standard OGSI interfaces:
  - GridService, HandleResolver, Factory, Notification, ServiceGroup
- GridService and Factory implementations are fundamental to GT3 container
  - Not easily replaced but could be extended
- Other interface implementations easily replaced if not adequate
- API and tools for creating new GridServices

Programming with GT3 java implementation

Server side:

- simplest way to write a Grid service is to extend GridServiceImpl
  - Tied to GT3 container
  - Locks functionality known at development time
  - Hard to port to other OGSI-compliant containers
- Dynamic delegation approach (operation providers)
  - Additional functionality can be added at deployment and runtime
  - Favours more modular, uncoupled, reusable designs
  - A little more code and a few lines in deployment descriptor

Inheritance and delegation examples

- Inheritance:
  - public class ExampleImpl extends GridServiceImpl implements ExamplePortType { … }
- Delegation:
  - **public class ExampleProvider implements OperationProvider** {

```
// Operation provider properties
Private static final QName[] operations =
        new QName[]{new QName("", "*")};
private GridServiceBase base;
// Operation Provider methods
public void initialize(GridServiceBase base) throws
        GridServiceException { this.base = base; }
public QName[] getOperations() { return operations; }
… }
```

  - **<parameter name="instance-className"**
    value="...providers.wsdl.ExamplePortType"/>
    <parameter name="instance-operationProviders"
    value="...providers.impl.ExampleProvider"/>
    <parameter name="instance-baseClassName"
    value="org.globus.ogsa.impl.ogsi.GridServiceImpl"/>

Programming with GT3 java implementation ...

Client side:

- A grid service client could be written directly on top JAX-RPC client API for Web services
  - JAX-RPC has no knowledge of GSH's and GSR's
  - GT3 provides a number of utility classes to simplify GSH to GSR resolution
- Extended JAX-RPC
  - GT3 provides custom stub generator extending the JAX-RPC stubs to integrate these utilities
    - Client gets a GSH to a service instance
    - GSH passed to a ServiceGridLocator that constructs a proxy responsible for making the call using binding defined in WSDL
    - Proxy exposed using standard JAX-RPC generated PortType interface

Client example

```
//Create a new ExampleService instance using factory exampleFactory
LocatorType locator = exampleFactory.createService();
//get a reference to its ExamplePortType
ExampleServiceGridLocator exampleLocator = new
                                    ExampleServiceGridLocator();
ExamplePortType example = exampleLocator.getExampleService(locator);

// Call remote method ‘doSomething()'

example.doSomething();
```

Tools, logging and testing

- Apache Axis, Java COG kit, jakarta ant, browsers
- Logging:
  - based on jakarta commons logging architecture
  - GT3 also provides a logging Grid service: OgsiLogging
- Testing:
  - Build on top of JUnit test toolkit
  - Test suits are named following a convention based on when they should be run and what kind of functionality they test
  - Tests can be run against a local server or a remote server

Designing a Grid Service

- A grid service is a web service that must implement the GridService portType in addition to its own portTypes.
  - Uses WSDL (or GWSDL) for interface definition
  - Uses standard Web services binding: encoding message transmission protocols (SOAP over HTTP in GT3)
  - GridService portType
    - Querying and updating against the serviceData set of the Grid service instance (serviceData mandated by OGSI and user defined)
    - Managing the termination of the Grid service instance
- No restriction on implementation of interfaces
  - Any language, any OS, any platform
  - Any internal state management

Designing a R-GMA schema Grid Service

- Design Constraints:
  - Backward compatible with existing R-GMA
  - Interchangeable with existing servlets
- One to one relationship between servlet and Grid service
  - expose schema API public methods in a WSDL portType
  - Use current implementation within a minimal Grid service (implementing GridService portType)
- Main change being network communication: SOAP over HTTP, instead of, HTTP get/post + XML string

Schema Grid Service implementation

- Start with JAVA interface

```
public interface Schema {
    public int    translateTableName(String tableName);
}
```

- Generate stubs
- Implement server

```
public class Schema extends PersistentGridServiceImpl implements
    SchemaPortType {
  …
  mySchema = new org.edg.info.SchemaInstance();
  …
  public int translateTableName(String tableName) throws RemoteException {
    …
    int answer = mySchema.translateTableName(tableName);
    return answer;
  }
}
```

- Write deployment descriptor
- Deploy service into container
- Write client and invoke service

Serializing ResultSet across network boundaries

- GT3 uses Apache Axis (as a SOAP engine)
- AXIS can send (via SOAP) basic types and arrays
- AXIS cannot send arbitrary classes without a registered AXIS serializer/de-serializer
- AXIS includes a serializer/de-serializer for java classes that follow the JavaBean pattern
- Our aproach:
  - Wrap ResultSet into a ResultSetBean, serialize it, and unwrap it at the other end
  - Start from XSD definition and use tools to generate the JavaBean
  - Provide the wrapper class

Serializing RGMAException across network boundaries

- OGSI recomments the use of WSDL:Fault by defining a base XSD type (ogsi:FaultType) from which all faults derive
- In AXIS
  - Java.rmi.RemoteException maps to SOAP:Fault element – useful if recipient able to create instance of received fault.
  - Other Exceptions represented as WSDL:fault elements – useful for new grid service implementation
- our approach: wrap RGMAException within RemoteException
  - RGMAException caught on service side
  - sent as cause of org.globus.ogsa.GridServiceException,
  - caught at other end
  - which then throws a new RGMAException having the caught exception as cause
  - Limitation: additional fields in RGMAException ignored.

Unresolved issues

- Clients in other languages
- Security- in java and other languages
- Producer/Consumer
- Replication
- How to incorporate OGSI features – ie: going beyond wrapping existing code and functionality