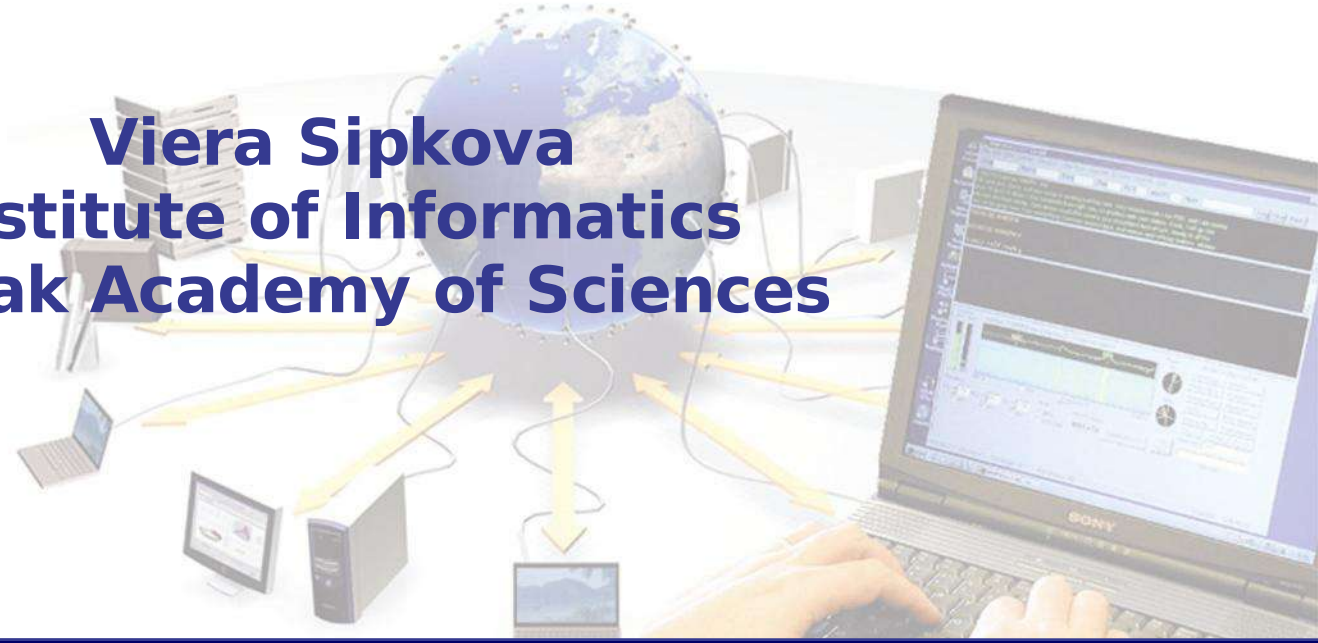
Enabling Grids for E-science in Europe

www.eu-egee.org

Grid Services and GT3

Viera Sipkova
Institute of Informatics
Slovak Academy of Sciences



Egee is a project funded by the European Union under contract INFSO-RI-508833

Grid Services and GT3

Enabling Grids for E-science in Europe

- **Outline**
- What is the Grid Service
- How to build a Grid Service
- How to access a Grid Service
- Future

Grid Services

Enabling Grids for E-science in Europe

A **Grid Service** is a Web Service
which conforms to a set of conventions
(interfaces and behavior) that define how
a client interacts with a Grid Service.

(I.Foster, C.Kesselman, S.Tuecke, J.M.Nick)

Web Services

Enabling Grids for E-science in Europe

- **Web Services** - the basis for Grid Services
  - A distributed computing paradigm, a technology for service oriented architectures - loosely coupled Client-Server applications.
  - Network accessible functions that can be invoked via a well-defined remote interface.
  - An interface for Web services is defined using the Web Services Description Language (WSDL).
  - The most common implementation of Web services works in a simple *Request-Response* principle (relying on SOAP and HTTP).

Key Concepts

Enabling Grids for E-science in Europe

- **Grid Services** - standard Web Services plus extensions.
- Grid Services are defined by the Open Grid Service Architecture (OGSA).
- Grid Services are specified by the Open Grid Services Infrastructure (OGSI).
- Globus Toolkit 3 (GT3) is an implementation of everything what is specified in OGSI, and therefore everything what is defined in OGSA.

Open Grid Services Architecture

- The **OGSA**, developed by the Global Grid Forum (http://www.ggf.org), defines a common, standard, and open architecture for grid-based applications - a distributed system framework based on OGSI.
- The Goal of the OGSA is to standardize all the services one finds in a grid application by specifying a set of standard interfaces for these services.
- A Grid Service interface corresponds to portType in the Web Services Description Language (WSDL).

Open Grid Services Infrastructure

Enabling Grids for E-science in Europe

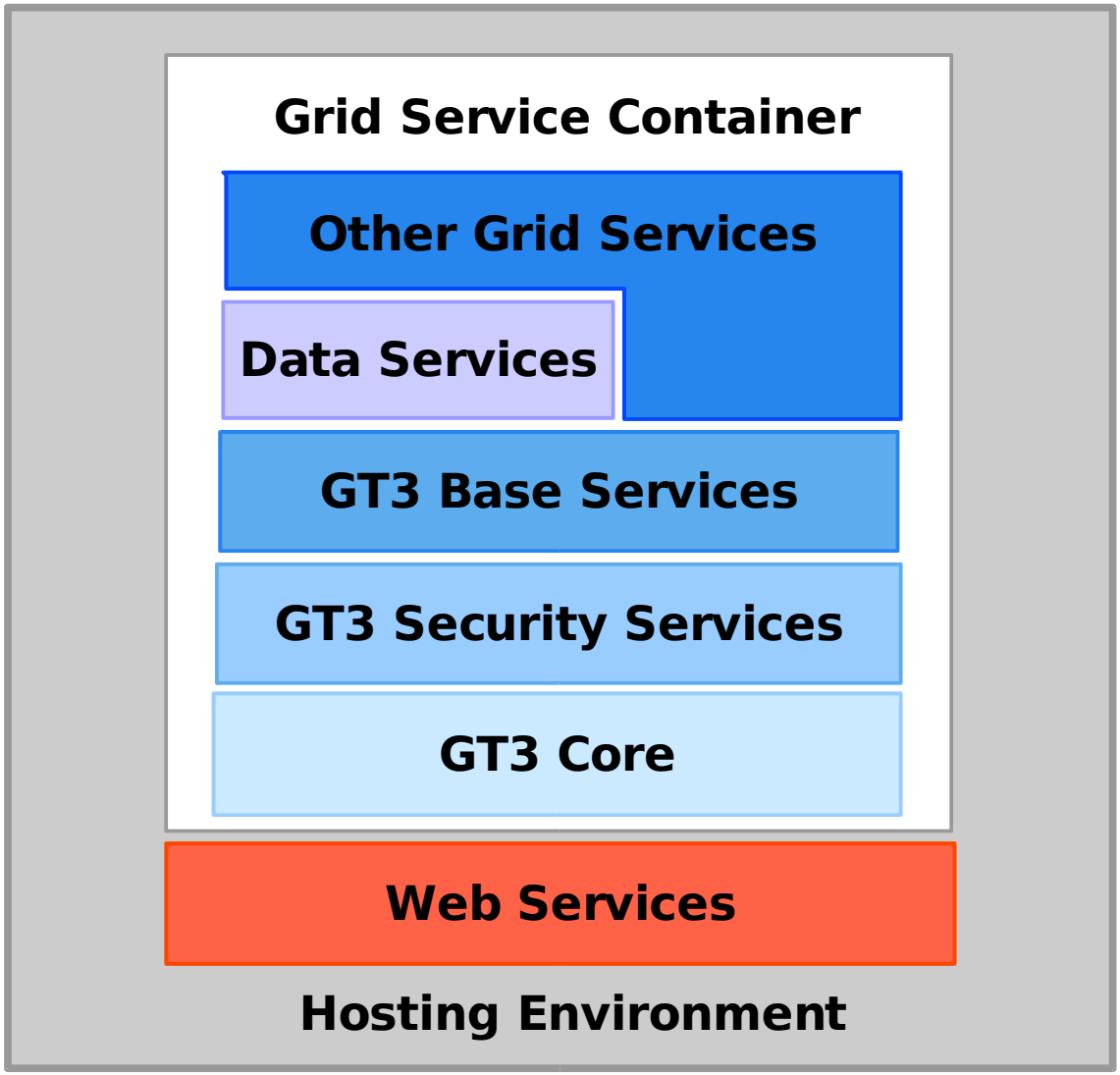
- The **OGSI**, developed by the Global Grid Forum (http://www.ggf.org), gives a formal and technical specification of what a Grid Service is and how it works.
- The **OGSI** defines mechanisms for creating, managing, and exchanging information among Grid Services.
- The **OGSI** represents the **Core** of the **GT3**, which provides a development support for exposing and accessing Grid Service implementations.

Globus Toolkit® 3

- The **Globus Toolkit 3 (GT3)**, developed by the Globus Alliance (http://www.globus.org), is a software toolkit that allows to construct Grid enabled tools, services, and applications.

- The **GT3** includes the complete implementation of OGSI – GT3 core, and a lot of other services, programs, and utilities.

Globus Toolkit® 3 Architecture



Globus Toolkit® 3 Components

- **GT3 Core**
  - Implementation of the OGSI specification
- **GT3 Base Services**
  - Index Service: allows to find the location of the Grid Service
  - Managed Job Service: allows to control the jobs
  - Reliable File Transfer Service: allows to transfer data files reliable
  - System Services: provide information about hosting environment (Ping, Logging)

Globus Toolkit® 3 Components

- **GT3 Security Services**
  - Several layers of security (SSL, X.509 certificates,...)
- **GT3 Data Services**
  - includes Replica Management
- **Other Grid Services**
  - non-GT3 services running on top of the GT3 architecture

*** GT3 Core ***

(OGSI Implementation)

- **OGSI Extensions**
  (in compare with standard Web Services)

  - Stateful and potentially transient services using a Factory/Instance model

  - Lifetime management
    – Grid Service instances can be created with a specified lifetime (Min and Max).
    – The initial lifetime can be extended by explicit request of the client.
    – Grid Service instances can be destroyed at any time.

- **OGSI Extensions** (cont.)

  - Grid Service Handle (GSH) & Grid Service Reference (GSR)
    – network-wide pointers which make accessible the Grid Service instances to client applications.

    GSH – the permanent Grid Service URI
    GSR – WSDL document resolved from the GSH
    GSR contains all information required to communicate with the service instance.

- **OGSI Extensions** (cont.)
  - Service Data
    allows to include a set of structured data to any Grid Service.
    Two categories of Service Data:
    – State Information (operation results, runtime information, etc.)
    – Service Metadata (system data, supported interfaces, cost of the service, etc.)

- **OGSI Extensions** (cont.)
  - Notification
    allows clients to be notified of changes occurring in a Grid Service.

    – Pull approach - the observable only informs the observers that a change has occurred.
    – Push approach - allows data to travel with the notification

- **OGSI Extensions** (cont.)
  - Logging
    allows writing a log of interesting events to the console or to a file.
    It is based on the Apache Jakarta Commons Logging component .
    6 levels: Debug, Trace, Info, Warn, Error, Fatal
  - Service Groups
    allows to group different services together and access them through a single point of entry.

Creating a Grid Service

1. **Define the Grid Service Interface** (with GWSDL)

2. **Implement the Grid Service** (with Java)

3. **Deploy the Grid Service**
   - **Configure** (with WSDD)
   - **Build** (with Ant)
   - **Deploy** (with Ant)

Creating a Grid Service

(1) Define the Grid Service Interface

- Grid Service interface is described through the Web Service Description Language (WSDL)
- Grid Service interface can be written in
  - Java (! some complex data types do not map very well into WDSL)
  - GWSDL (extended WSDL 1.1) - using the GridService portType definition

(1) Define the Grid Service Interface (cont.)

- Grid Service interface is converted into the WSDL form through some Ant build tools (GWSDL2WSDL, generateWSDL, generateBinding,...).
- From the WSDL interface file --> Java stubs to handle the requests and forwarding them to the service implementation.

Creating a Grid Service

- Example: **Grid Service Interface in Java**

```
public interface Math
{
   public void add(int a);
   public int   getValue();
}
```

Creating a Grid Service

- Example: **Grid Service Interface in GWSDL**

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions  name="MathService"
  targetNamespace=
      "http://www.globus.org/.../MathService"
   xmlns:tns="http://www.globus.org/.../MathService"
   xmlns:ogsi="http://www.gridforum.org/.../OGSI"
   xmlns:gwsdl=
      "http://www.gridforum.org/.../gridWSDLExtensions"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   ...>

<import location="../../ogsi/ogsi.gwsdl"
   namespace="http://www.gridforum.org/.../OGSI" />
...
</definitions>
```

Creating a Grid Service

- Example:
  Grid Service Interface in GWSDL (cont.)

```
<gwsdl:portType name="MathPortType"
        extends="ogsi:GridService">

   <operation name="add">
      <input message="tns:AddInputMessage" />
      <output message="tns:AddOutputMessage" />
      <fault name="Fault" message="ogsi:FaultMessage" />
   </operation>

  <operation name="getValue">
      ...
   </operation>
</gwsdl:portType>
```

Creating a Grid Service

- Example:
  Grid Service Interface in GWSDL (cont.)

```
<message name="AddInputMessage" >
   <part name=parameters" element="tns:add" />
</message>
<message name="AddOutputMessage" >
   <part name=parameters" element="tns:addResponse" />
</message>

<message name="GetValueInputMessage" >
   ...
<message name="GetValueOutputMessage" >
   ...
```

Creating a Grid Service

- Example:
  Grid Service Interface in GWSDL (cont.)

```
<types>    <!-- XML Schema type model -->
<xsd:schema  ...>

  <xsd:element name="add">
   <xsd:complexType> <xsd:sequence>
     <xsd:element name="value"  type="xsd:int" />
   </xsd:sequence> </xsd:complexType>
  </xsd:element>
  <xsd:element name="addResponse">
   <xsd:complexType />
  </xsd:element>
   ...
<xsd:schema>
</types>
```

Creating a Grid Service

(1) Define the Grid Service Interface

(2) Implement the Grid Service

- Implementation by inheritance
  One Java class – extension of the skeleton class GridServiceImpl (implementation of the GridService portType)

- Implementation by delegation (Operation Providers)
  Several Java classes – created by implementing the OperationProvider

Creating a Grid Service

- Example: **Implementation by Inheritance**

```
package org.globus.examples.services.first.impl;

import org.globus.ogsa.impl.ogsi.GridServiceImpl;
import org.globus.examples.stubs.MathService.MathPortType;
import java.rmi.RemoteException;

public class MathImpl
    extends GridServiceImpl implements MathPortType
{
   ...
}
```

Creating a Grid Service

- Example: **Implementation by Inheritance** (cont.)

```
public class MathImpl
        extends GridServiceImpl implements MathPortType
{
   private int value=0;

   public MathImpl()
   {  super("Simple MathService");  }

   public void add(int a) throws RemoteException
   {  value = value + a;  }

   public int getValue() throws RemoteException
   {  return value; }
}
```

Creating a Grid Service

Enabling Grids for E-science in Europe

1. Define the Grid Service Interface

2. Implement the Grid Service

3. **Deploy the Grid Service**
   - **Configure** (with WSDD)
   - **Build** (with Ant)
   - **Deploy** (with Ant)

Creating a Grid Service

(1) Define the Grid Service Interface

(2) Implement the Grid Service

(3) Deploy the Grid Service

- **Configure** the service using the deployment descriptor

- Deployment Descriptor
  - defines how the Grid Service should be published
  - is written in Web Service Deployment Descriptor (WSDD) format

Creating a Grid Service

- Example: **Deployment Descriptor in WSDD**

```
<?xml version="1.0">
<deployment name="defaultServerConfig"
   xmlns="http://xml.apache.org/axis/wsdd">
  <service  name="examples/first/MathService"  ...>
     <parameter  name="name" value="MathService"  />
     <parameter  name="className" value="org.globus..." />
     <parameter  name="baseClassName" value="org..." />
     <parameter  name="schemaPath" value="schema/... />
     ...
     <parameter  name="persistent" value="true" />
     ...
   </service>
</deployment>
```

Creating a Grid Service

(1) Define the Grid Service Interface

(2) Implement the Grid Service

(3) Deploy the Grid Service

- Configure the service
- **Build** the grid service binary (GAR file)

- Grid Archive (GAR) file
  - includes all files and information needed to install the Grid Service in the runtime environment
  - is created by the Ant build tool

Creating GAR file with Ant

- **Ant build tool** (http://jakarta.apache.org/ant)

1. Converts the GWSDL interface into WSDL
2. Creates stubs classes from the WSDL
3. Compiles the stubs classes
4. Compiles the service implementation
5. Organizes all the files into a very specific directory structure (GAR file)

Creating a Grid Service

(1) Define the Grid Service Interface

(2) Implement the Grid Service

(3) Deploy the Grid Service

- Configure the service
- Build the service
- **Deploy** the service into the runtime environment (Grid Service container)

- Deployment is done with the Ant build tool

Deployment with Ant

Enabling Grids for E-science in Europe

- **Ant build tool**
  - Unpacks the GAR file and copies the files within (WSDL, WSDD, compiled stubs, and compiled implementation) into key locations in the GT3 directory tree.
  - According to the deployment descriptor it configures the web server to take the new Grid Service into account.

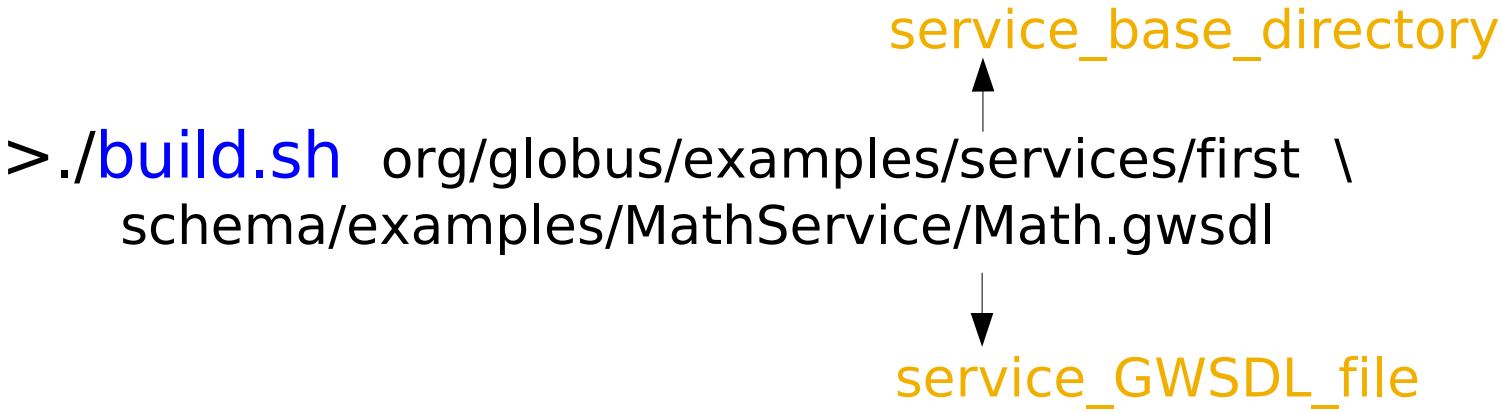
Input Files for Ant

- **Source files**
  - Grid Service interface (GWSDL)
  - Grid Service implementation (Java)
  - Deployment descriptor (WSDD)
- **Build files**
  - Build script - make file
  - Ant build file - directs Ant what to do and how
  - Namespace mappings file - maps GWSDL namespaces to Java packages
  - GT3 build files

Grid Service Directory Structure

```
gt3_examples/   --> build files  (build.sh, build.xml,
  |                                 namespace2package.mappings)
  |--  schema/
  |      |-- examples/   --> GWSDL files and XML Schema files
  |                            (Math.gwsdl)
  |-- org/
  |    |-- globus/
  |          |-- examples/
  |                |-- services/   --> Service implementation files
  |                |     |-- first/
  |                |     |      |-- server-deploy.wsdd   --> Deployment descriptor
  |                |     |      |-- impl/                --> Implementation classes
  |                |     |                                   (Math.java)
  |                |     |-- ...
  |                |-- clients/  --> Client implementation files
  |
  |-- build/   --> Files generated with Ant
                   (lib/org.globus.examples.services.first.gar, ...)
```

Grid Service Building Process




```
>cd $GLOBUS_LOCATION
>ant deploy -Dgar.name= \
  ~/gt3_examples/build/lib/org.globus.examples.services.first.gar
```

```
>globus-start-container
```

(Output: a list of deployed services)

Creating a Client

- **Access to a Grid Service**

- Client side infrastructure model: **Stubs** - a common approach enabling client access to Grid Services.
- Stubs are automatically generated when building the Grid Service.

- A client gains access to a Grid Service through the Grid Service Handle and Grid Service Reference. OGSI provides mechanisms (HandleResolver) to support client resolution of a GSH into the GSR.

Simple Client

- Example: **Command-line Client**

```
package org.globus.examples.clients.MathService;

import ...stubs.MathService.service.MathServiceGridLocator;
import ...stubs.MathService.MathPortType;
import java.net.URL;

public class MathClient
{  public static void main(String[] args)
   {
     ...
    }
}
```

Simple Client

- Example: **Command-line Client** (cont.)

```
public class MathClient
{  public static void main(String[] args)
   {   try {
       // Get command-line arguments
       URL GSH = new java.net.URL(args[0]);
       int a = Integer.parseInt(args[1]);

       // Get a reference to the MathService instance
       MathServiceGridLocator mathSL =
          new  MathServiceGridLocator();
       MathPortType math=mathSL.getMathServicePort(GSH);
       ...
       }
   }
}
```

Simple Client

- Example: **Command-line Client** (cont.)

```
public class MathClient
{  public static void main(String[] args)
   {  try {
      ...
      // Call remote method 'add'
      math.add(a);
      System.out.println("Added " + a);

      // Get current value through remote method 'getValue'
      int value = math.getValue();
      System.out.println("Current value: " + value);

      } catch(Exception e)
      { ... }
   }
}
```

Creating a Client

- **Building the Client**

```
>javac -classpath ./build/classes/:$CLASSPATH \
        .../examples/clients/MathService/MathClient.java
```

- **Running the Client**

```
>java -classpath ./build/classes/:$CLASSPATH  \
        ...examples.clients.MathService.MathClient       \
        http://147.213.65.242:8080/ogsa/.../MathService  \
        5
```

- **Output**

```
Added 5
Current value: 5
```

Summary

- **GT3-OGSI**
  - The OGSI specification is long and dense.
  - OGSI does not work well with current Web Services tooling.
  - OGSI is too object oriented.

- **Next Future**

➔ **OGSI will converge with Web Services standards.**

Next Future

Enabling Grids for E-science in Europe

- **Web Services Resource Framework** (WSRF) & **Globus Toolkit 4** (implementation of WSRF)
  - In January 2005 a new standard will be available to substitute OGSI.
  - OGSA will be based directly on Web Services instead of being based on OGSI Grid Services.
  - High level Grid Services defined in OGSA will keep having the same interfaces and behavior, only the underlying middleware will be pure Web Services.

Final

Enabling Grids for E-science in Europe

- **Global Grid Forum**
  http://www.ggf.org

- **Globus Alliance**
  http://www.globus.org

- **Apache software**
  http://jakarta.apache.org
  http://xml.apache.org
  http://ws.apache.org

- **Thank you for your attention.**