Enabling Grids for E-science in Europe

www.eu-egee.org

JRA1 all-hands-meeting, Padua 15.11.2004

Configuration & Management for gLite

Joachim Flammer
Integration Team

EGEE is a project funded by the European Union under contract IST-2003-508833

Contents

- Getting to the GRID
- Management aspects
- Configuration aspects
  - Storing of configuration
  - Accessing configuration
  - Updating configuration
- The global picture
- Summary

Getting to the GRID

Software manageability

If we want to get gLite going on a large scale we need manageable software.

- What is manageability?
  - Exercise administrative and supervisory actions and receive information that is relevant to such actions
- Manageability is made of
  - **Monitoring**
  - **Tracking**
  - **Control**
    - Ability to alter runtime behavior of a managed component

The aim of the EGEE project is to build on recent advances in grid technology and develop a service grid infrastructure in Europe which is available to scientists 24 hours-a-day.

Why configuration is important

A quote from Zdenek Sekera (SA1) (on reply to the first gLite Component)

[…]
Configuration
One the ***very major requirements*** *for the developers, which we discussed on the SA1-JRA1 meeting very early at the project was to make developers aware of the fact that* ***one the most damaging issues for the EDG software was the extreme complexity of the configuration****.*
At that time we suggested to have ***one configuration file for all services****, split in 3 parts, first what the site has to change, second what it can change if enough knowledge is around and third what it should never dare to change unless under machine gun. […]*
I can accept the temporary way of doing things, unfortunately, ***I know for the fact that the temporary will very quickly turn permanent because developers find always something more important to do****. How is this problem to be handled?* ***I suggest we both consider this as an extremely important part of the software, and all means must be taken from the very beginning to ensure we don't finish EDG-way.***
Comments?

Mine: SA1 will be uncompromising on this issue.

[…]

Configuration

- General aspects
  - We have to get to a corporate approach for configuration in
    - Storing the values
    - Accessing the values
      - Human/Management side
      - Application side
    - Changing the values
  - The system has to be manageable
  - User friendliness: keep the end user in mind
  - Modular approach
    - Separate areas
    - Allow for changes later – new technologies etc.
    - Different plug-ins

Storing the configuration

- Use a unique way of storing the configuration
  - one type of configuration files
  - Combine
    - As human readable as possible
    - Properly formatted
  - **Our proposal: XML** ➔ *see Robert's talk*
- Collect configuration information at a single place
  - Set of files
  - Database
- Apply hierarchical structure for configuration information
  - Group configuration according their scope
    - gLite wide versus service configuration
    - machine versus user configurations
    - Values that should, may, cannot be changed
  - **Our proposal: One file for global values, one file per service** ➔ *see Robert's talk*
- Validate configuration
  - Type, range ➔ *see Robert's talk*
- Support user in understanding and selecting the correct value
  - Description of configuration value
  - Default values
  - Example values ➔ *see Robert's talk*

Configuration types

- We foresee to have two configuration value types:
  - Configuration values that can changed **dynamically** in a running application without stopping/restarting the application
    - Examples
      - log levels
      - allowed number of connections in a DB pool
      - …
  - **Static** configuration values that cannot be changed dynamically and that need the application to be restarted
    - Examples
      - Database connections
      - Hostname
      - Ports
      - ….

As services depend on each other, we should opt for as much as possible "dynamic" configuration values so that other applications are disturbed as little as possible !!!

Accessing the configuration

- Provide easy access of the configuration values by providing APIs
  - **Manager/Human**
    - Manager should have **one common place** (management console) to look and change the configuration
    - Manager of system **does not need to know** what needs to be reconfigured/restarted
  - **Application**
    - **one common interface**
    - language and platform depending implementations
    - All services should have the **same implementation** for a given language/platform

      ➔ *see Robert and Paolo's presentation for C++ and mine for Java*
    - **Common approach** to read/update configuration values from applications

Changing the configuration

- Controlled change of configuration
  - Manager changes configuration via a common management console
  - Changing the configuration should automatically trigger the necessary actions
    - Reconfigure applications dynamically for dynamic configuration values
    - Restart applications for static configuration values
    - Take into account the dependencies between applications
      - e.g. to restart tomcat the web services have to be restarted as well
    - Take into account the state of the application
      - an application wants to finish a transaction first
  - Track changes
    - What is the actual configuration of the application?

      ➔ Monitoring

The global picture

- We foresee a modular structure
  - ServiceConfigurator
    - sits inside the service and is the unique interface to
      - access configuration
      - get configuration values (e.g. for monitoring)
      - give common functionalities (e.g. start/stop) not implemented by the container
  - gLiteConfigurator
    - global tool to
      - manipulate configuration (management console)
      - start/stop services
      - ...

First step is getting the ServiceConfigurator going …

The next steps

- Many aspects not covered so far
  - Backup of values
  - Relation between services
  - …
- Time until the first release is limited
  - we have to get a working solution fast on which we can build on
  - If this is working we can add to the framework
  - Keep modular approach in mind

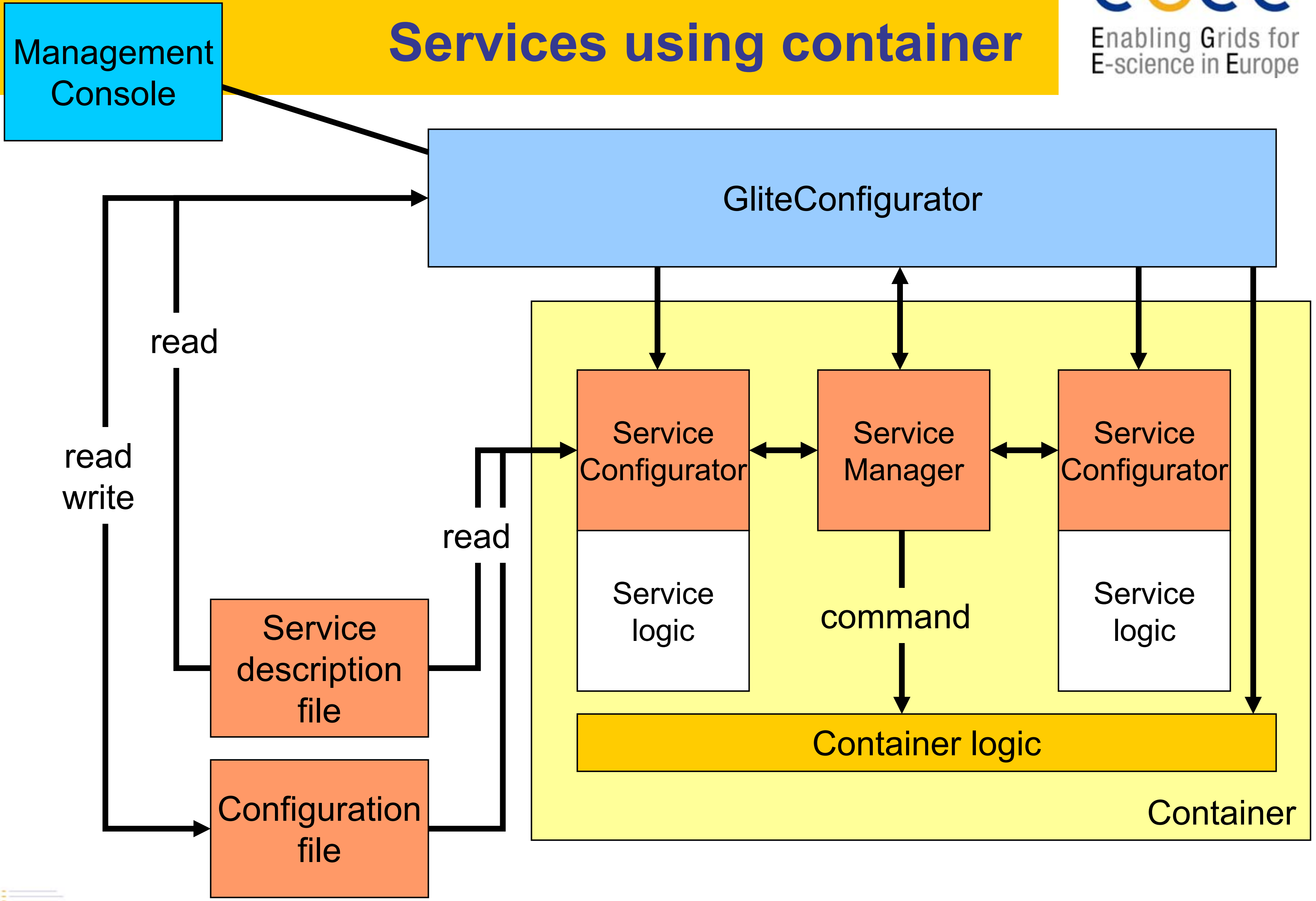
Summary

Enabling Grids for E-science in Europe

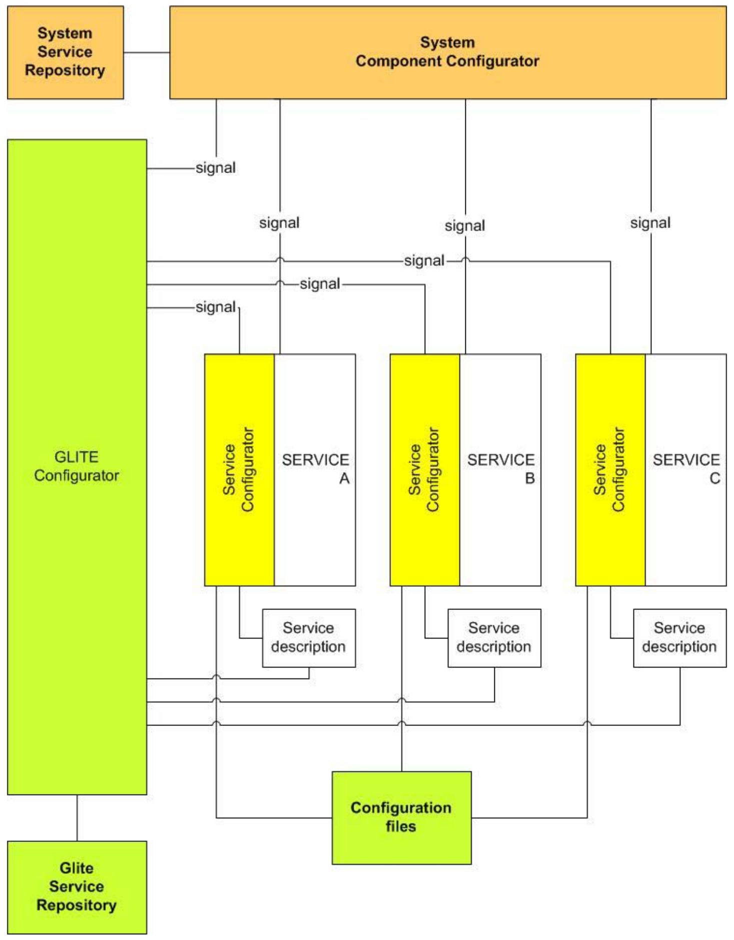
- Manageability is one of the key corner stone for gLite
- Manageability concerns different aspects that are interconnected: Monitoring – Tracking – Control
- Key aspect of configuration:
  - Common approach
  - Modular structure
- Configuration has to tackle
  - Storing of configuration values
  - Accessing configuration values
  - Update of configuration values
- Many more stuff to be considere uture

Build the foundation now !

Extra slide: Services using container



Extra slide: Old schema


System Service Repository
System Component Configurator
signal
signal
signal
signal
signal
signal
signal
GLITE Configurator
Service Configurator
SERVICE A
Service Configurator
SERVICE B
Service Configurator
SERVICE C
Service description
Service description
Service description
Configuration files
Glite Service Repository