



DOCUMENTATION OF THE FIRST PROTOTYPE - PORTALS AND ROAMING ACCESS FINAL VERSION

WP3.1

Document Filename:	CG3.1-D3.3-v1.0-PSNC030-PrototypeDoc.doc
Work package:	WP3.1
Partner(s):	ALGO, DATAMAT, PSNC
Lead Partner:	PSNC
Config ID:	CG3.1-D3.3-v1.0-PSNC030-PrototypeDoc
Document classification:	PUBLIC

Abstract: This document describes the first prototype implemented within the confines of the task entitled "Portals and Roaming Access".

Delivery Slip

	Name	Partner	Date	Signature
From				
Verified by				
Approved by				

Document Log

Version	Date	Summary of changes	Author
1-0-DRAFT-A	12/1/2003	Draft version	Rafał Lichwała, Bartosz Palak, Marcin Płóciennik, Paweł Wolniewicz, Mirosław Kupczyk, Norbert Meyer, Stefano Beco, Marco Sottilaro, Yannis Perros, Angelos Sphiris

CONTENTS

EXECUTIVE SUMMARY	4
1. INTRODUCTION	5
1.1. PURPOSE	5
1.2. DEFINITIONS, ABBREVIATIONS, ACRONYMS	5
2. REFERENCES	7
2.1. SOURCE CODE	7
2.2. CONTACT INFORMATION	7
3. IMPLEMENTATION STRUCTURE	8
3.1. MIGRATING DESKTOP	8
3.2. APPLICATION PORTAL	10
4. PROTOTYPE FUNCTIONALITY	11
5. USER MANUAL	12
5.1. DEPENDENCIES	12
5.2. INSTALLATION	12
5.3. RUNNING	12
5.4. INTERFACE DESCRIPTION	12
6. INTERNAL TESTS	13
7. ISSUES	14
8. APPENDIXES	15
8.1. FREQUENTLY ASKED QUESTIONS	15
8.2. DESCRIPTION OF APPLICATION CONTAINER AND PLUGIN	16
8.2.1. <i>General description of application container and application plugin</i>	16
8.2.2. <i>Preparation and execution of the Application Plugin</i>	20

EXECUTIVE SUMMARY

The purpose of this document is:

- to define the functionality that will be available for the first prototype of the Migrating Desktop and underlying modules (Roaming Access Server), as well as the Application Portal.

To show the current implementation state;

- To show the way of installing, using and testing WP3.1 modules
- To list the functionality, which is still required from other Work Packages for the first prototype and future development.
- To point out all issues, which affect the functioning of the prototype, including any known bugs

To make the structure of deliverable not that complicated and easy to use. It will be delivered as a set of standalone documents. It will consist of:

- CG3.1-D3.3-v1.x-PSNC0xx-PrototypeDoc – the main document of the deliverable;
- CG3.1-D3.3-v1.x-PSNC0xx-PrototypeInstallationGuide – installation guide;
- CG3.1-D3.3-v1.x-PSNC0xx-PrototypeFunctionality – first prototype functionality description;
- CG3.1-D3.3-v1.x-PSNC0xx-PrototypeUserGuide – user guide;
- CG3.1-D3.3-v1.x-PSNC0xx-PrototypeTests – description of tests procedures;

1. INTRODUCTION

Deliverable 3.3 is prepared respecting the Crossgrid Milestone month 12th. It is when the first public Prototype of the Portal and Roaming Access utility will be presented. WP 3.1 developers will put forward the Migrating Desktop and Portals. The Migrating Desktop is an advanced GUI for CrossGrid resources respecting administrative boundaries, security policy, etc. The Portal is implemented as a thin client accessible through the web browser only. The proposed solutions have advantages and disadvantages. Depending on the mobile user requirements, the user should make a trade off choosing the fat client respecting the time of the Migrating Desktop downloading. Then the work is more comfortable, but if the user chooses the Portal, he will obtain a limited but faster user interface.

1.1. PURPOSE

The purpose of the 1st Prototype is gathering all consistent information respecting proper integration of the dependent modules. As a result of the correct integration procedure, the working piece of software should be delivered. We assume that the bundle of CrossGrid software is still under development and the prepared version is still marked as “under construction”.

For the prototype the following features and tools will be ready (often within the limited functionality):

- Handling the Graphical User Interface;
- User configuration management;
- Handling transmission of files;
- Job submission;
- Monitoring of the job execution;
- Managing local application and files;
- Managing remote files;

1.2. DEFINITIONS, ABBREVIATIONS, ACRONYMS

API	Application Program Interface
APS	Application Portal Server
CoG	Commodity Grid Toolkit
GUI-C	Graphic User Interface Container
HEP	High Energy Physics
HTTP	HyperText Transport Protocol
HTTPS	HyperText Transfer Protocol Secure
HTTP-S	HTTP Server
JDL	Job Description Language
JSP	Java Server Pages
LDAP	Lightweight Directory Access Protocol
MD	Migrating Desktop
PDA	Personal Digital Assistant

RA	Roaming Access
RAS	Roaming Access Server
SA	Scheduling Agent
SM	Session Manager
SOAP	Single Object Access Protocol
SOM	Self Organising Map
SSL	Secure Sockets Layer
SVG	Scalable Vector Graphic
TBD	To Be Defined
VO	Virtual Organisation
WB	Web Browser
XML	Extended Mark-up Language

2. REFERENCES

2.1. SOURCE CODE

The main repository source code is available at FZK. This is the primary and official CVS data base. It can be located at the following address:

http://gridportal.fzk.de/cgi-bin/viewcvs.cgi/crossgrid/crossgrid/wp3/wp3_1-portals/

(the working instance of the WP3.1 data is located at PSNC).

Detailed class descriptions (generated by JavaDoc tool) will be placed at the same location <http://ras.man.poznan.pl/javadocs>.

2.2. CONTACT INFORMATION

The contact person is Pawel Wolniewicz (PSNC), pawelw@man.poznan.pl.

3. IMPLEMENTATION STRUCTURE

3.1. MIGRATING DESKTOP

3.1.1. System decomposition

[based on “WP3.1 Software Requirements Specification” – chapter 2.1 and “WP3.1 Design Document” – chapter 4]

The main components of the proposed architecture of Task 3.1 and their position in the layered structure are shown on Fig. 1 and Fig. 2.

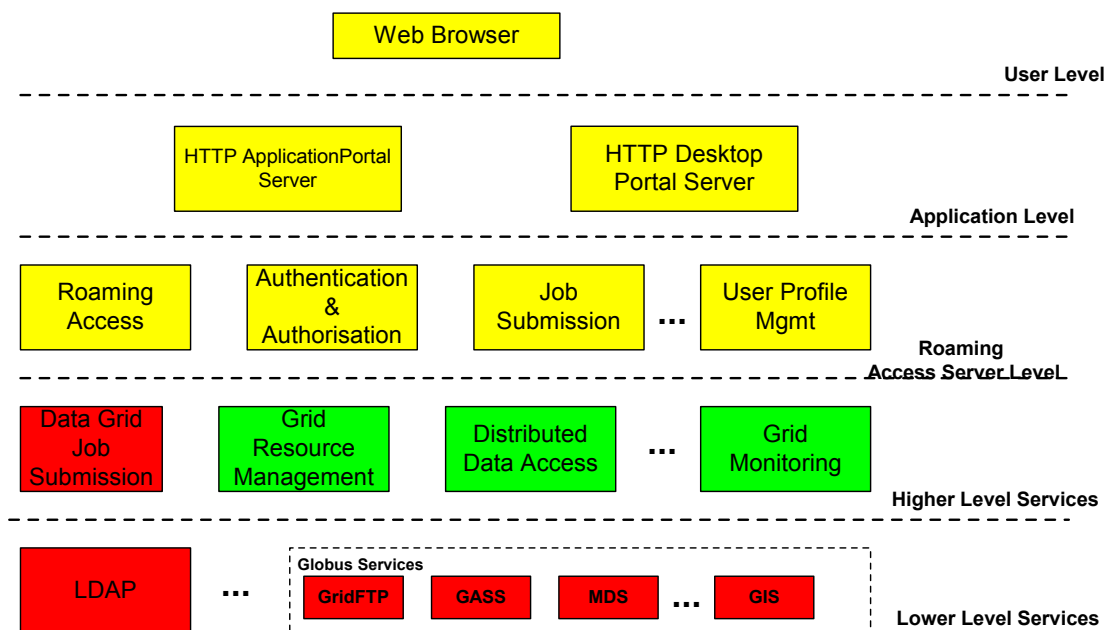


Fig. 1 Layered view of Task 3.1 architecture

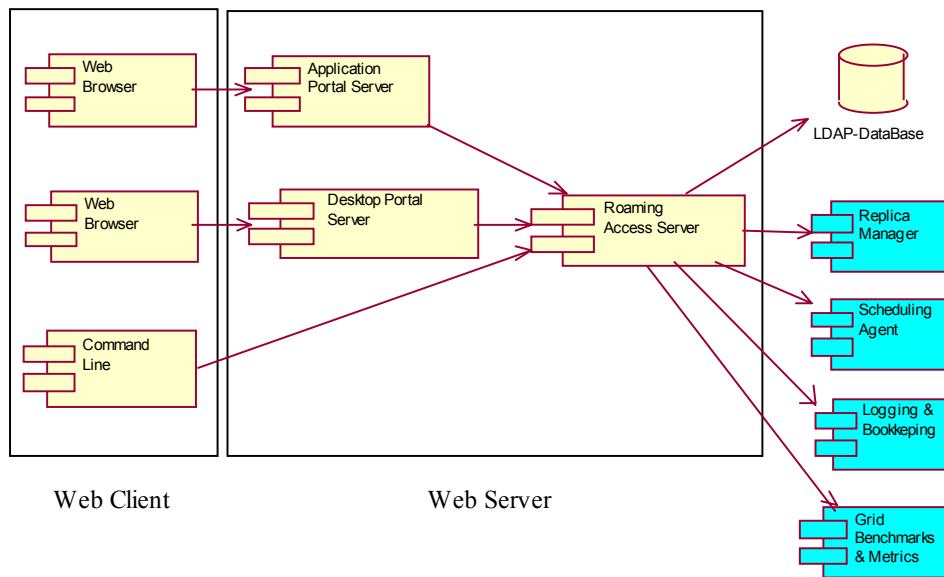


Fig. 2 Component view of Task 3.1

The main components of the proposed architecture were (see Fig. 1 and Fig. 2):

- Web Browser – a simple web browser displaying pages generated by HTTP Server or handling the Migrating Desktop applet;
- Command Line Interface – a tool that gives the user text interface to the Roaming Access Server (basic services only);
- Application Portal Server – a service that provides information for HTTP Server needed to create web pages. It keeps information about user sessions and provides first parameter verification (see description below);
- Desktop Portal Server – a service that extends the functionality of the Application Portal Server by providing a specialised advanced graphical user interface and a sharing mechanism that allows the user to make files stored on his machine available from other locations/machines;
- Roaming Access Server – a network server responsible for managing user profile, authorisation and authentication, job submission, file transfer and grid monitoring (see description below).

3.1.2. Implementation state

- Web Browser – is not a subject of WP3.1 work; Any web browser available on the market that supports java plugin (as e.g. Netscape, MS Internet Explorer, etc.) can be used for accessing portal pages or the Migrating Desktop applet;
- Command Line Interface – has no entered implementation phase;
- Application Portal Server – see chapter 3.2 for details;
- Desktop Portal Server – at that moment serves only the Migrating Desktop applet and all corresponding java archives; the mechanism for serving specialized advanced graphical user interface has been moved to the applet (web client) module, so the problem if the

Desktop Portal Server shall be a (logically or physically) separate module becomes an open question.

- Roaming Access Server – the main and most important module of Task 3.1 system. Several services (based on web-services mechanism) have been implemented:
 - managing user profile services – responsible for storing/restoring user configuration;
 - (limited) authentication mechanism – that gives simple access control;
 - job submission service – allows to submit grid application;
 - Virtual User Directory service – allows to access all user files stored in different physical locations as they were stored in one logical location; manages file transfer;
 - job monitoring – gives information about the current job state to the user; allows to cancel a job;
- Web Client – understood as loaded from web location java applet – a graphical user interface to services offered by the Roaming Access Server has been implemented;

3.1.3. Implementation model vs. the design model

Except for changes mentioned in chapter 3.1.2, the implementation of the Migrating Desktop modules proceeds according to the design model without remarkable discrepancies.

3.2. APPLICATION PORTAL

The Graphical User interface of the Application Portal Server has been largely implemented and provides links to operations that will be implemented later on, during the course of the project. These operations concern four CrossGrid applications as well as other modules, such as MPI verification and the running of benchmarks.

The user authentication mechanism has been implemented except for the use of certificates. This discrepancy is due to the fact that the details of the certificate handling mechanism have not been agreed upon at the time of writing.

A mechanism for storing and retrieving user-specific settings has also been implemented. These settings constitute what is known as a *portal user profile* and are the following:

- the theme, i.e. a colour scheme that determines how the portal's pages appear to the user
- the activation of a personal menu where the user may store a collection of links of personal use
- the determination of the content of the personal menu
- the determination of its position within a portal window

4. PROTOTYPE FUNCTIONALITY

See the attached document CG3.1-D3.3-v1.0-PSNC032-PrototypeFunctionality.doc which describes the first prototype functionality in details.

5. USER MANUAL

5.1. DEPENDENCIES

See the attached document CG3.1-D3.3-v1.0-PSNC031-PrototypeInstallationGuide.doc which contains a detailed list of software and hardware requirements and dependencies.

5.2. INSTALLATION

See the attached document CG3.1-D3.3-v1.0-PSNC031-PrototypeInstallationGuide.doc which describes the first prototype installation procedure in details.

5.3. RUNNING

See the attached document CG3.1-D3.3-v1.0-PSNC031-PrototypeUserGuide.doc which describes in details how the user can use the prototype.

5.4. INTERFACE DESCRIPTION

WP3.1 task, due to its high level position in The CrossGrid Project structure as the unified and consistent window to the CrossGrid environment, will provide no interfaces to external modules.

6. INTERNAL TESTS

See the document *CG3.5-D3.3-v1.0-CSIC021-TestIntegrationPrototype.doc* for the list of test procedures.

7. ISSUES

Known bugs:

- Loading data from some network location via web services takes too much time;
- Migrating Desktop GUI loaded to user workstation as Java applet works slowly;

8. APPENDIXES

8.1. FREQUENTLY ASKED QUESTIONS

Q: What is the difference between the Migrating Desktop and the Portal ?

A: The Portal provides functionality which is similar to the Migrating Desktop, but finally its look and user settings strongly depend on the web browser and operating system which are used by the user. As opposed to the Migrating Desktop, the Portal allows very limited user customisation of their work environment.

The Migrating Desktop provides much more sophisticated availability of customisation of user's profiles. It is similar to the desktop of graphical operating systems, which allows to manage windows and icons. The Migrating Desktop can be equalled with mobile devices like notebooks. User has his own preferred and customized workspace with all its software (Migrating Desktop tools), settings (user profiles) and data (documents on the desktop). Such "notebook" (the Migrating Desktop) can be treated as a tool for accessing and working with Grid environment and its resources without losing user's preferences.

Q: What is the difference between the Migrating Desktop and VNC (Virtual Network Computing) ?

A: VNC is a special kind of protocol which allows to work with X-based applications that are executed on other machines. It provides only the functionality for displaying a graphical representation of a given application. The user has to have an active connection with the VNC-server while working with the remote application (on-line work).

The Migrating Desktop is an advanced tool that allows the user to:

- authorize him in the Grid environment;
- use many dedicated tools for different aspects of the Grid;
- store and restore his preferences and settings for the graphical user interface;
- have access to high computing power machines and application connected with the Grid;

Q: Why does the Migrating Desktop work so slowly ?

A: The main principle of the Migrating Desktop is its mobility as well as software and hardware independence. This is the reason why the Migrating Desktop was implemented in Java language (independence) and was released as a Java Applet (mobility). On old and slow machines Java application works not so fast as native executable software, but this is a cost of its flexibility. The Migrating Desktop is still improved and optimised, but we must remember about the Java language architecture and the speed of network connection limitations that influence the Migrating Desktop efficiency.

Q: Why is fast Internet connection preferred while working with the Migrating Desktop ?

A: The Migrating Desktop can be used in off-line work when the user wants to prepare some parameters of the Grid jobs for future submission or to change some workspace

settings. Finally all those data have to be sent to the Roaming Access Server where the whole user's profile is stored. When the user starts the Migrating Desktop the whole application (Java applet) and the related data (information about user profile, a list of jobs, content of the user virtual home directory) have to be downloaded before its execution. This is the reason why the Internet connection should be fast enough (it is strongly recommended but not required to work with the Migrating Desktop). The faster the Internet connection, the more productivity of work with the Migrating Desktop we have.

Q: What are the advantages of working with the Migrating Desktop ?

A: The Migrating Desktop has the following features which make the user's work with the Grid environment easier and faster:

- full user profile restoring;
- availability of work with several Grid environment at the same time (for example with CrossGrid in one opened window and PROGRESS in another);
- a set of useful tools for interacting with Grid resources (GridFTP, JobSubmission, JobMonitoring etc.);
- a possibility of off-line work;

8.2. DESCRIPTION OF APPLICATION CONTAINER AND PLUGIN.

8.2.1. General description of application container and application plugin

Application plugin and container in general idea of showing application specific input and output. This will allow to prepare a portal framework that is independent of the application type so that we could easily extend it and add next applications.

The Portal and desktop provide a wizard that the user can use to specify job details. This wizard will simplify the process of specifying parameters and limits, suggesting user default or last used parameters. That wizard will consist of several panels. Two panels are reserved for application specific plugin – Arguments panel and Job Output panel. (The contents of those panels that mean “application plugin” will be implemented by applications)

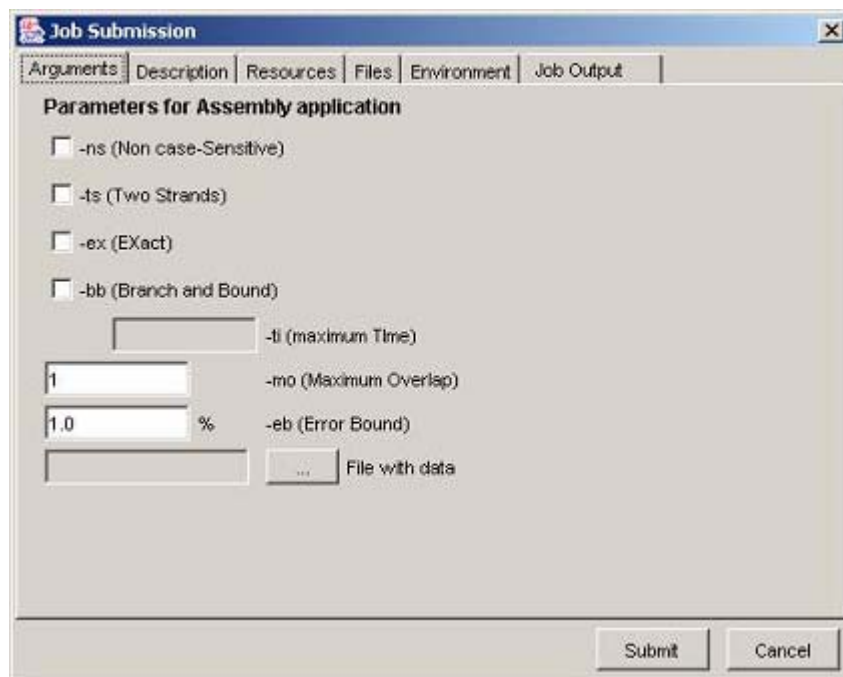


Fig. 3 Job Submission dialog – “Arguments” page

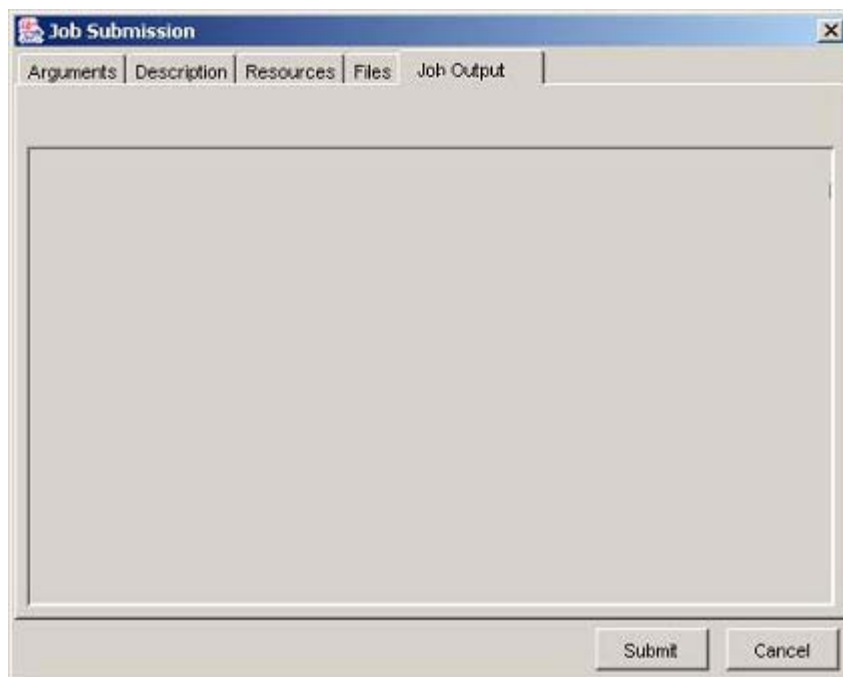


Fig. 4 Job Submission dialog – “Job Output” page

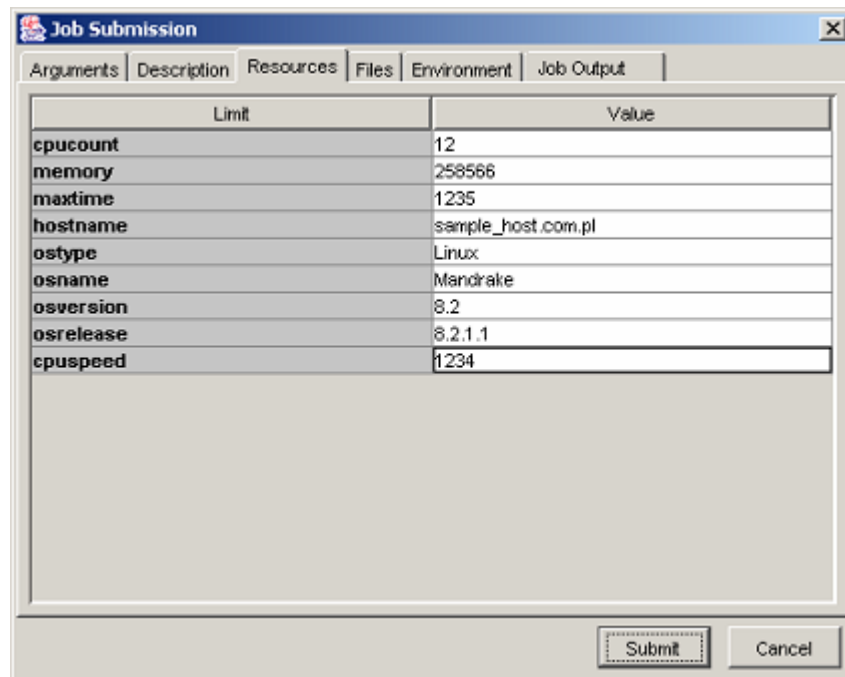


Fig. 5 Job Submission dialog – “Resources” page

The application container is a framework for application plugin.

- Frame that application plugin will appear
- It is application independent
- It will give ApplicationContainerAPI that can be accessed from plugin
- It can steer application plugin by restarting it, stopping, catching error messages and invoking appropriate actions
- It can set or get parameters within the application plugin

The application plugin is a content of the application container. It should provide and implement two panels of JobSubmissionWizard: arguments and job output, and should be able to visualise the output of application. The application plugin must implement ApplicationPluginInterface.

- It will have reference to the Application Container API, so that it could invoke some function from the application container if necessary.
- It can be on a different server than the Application Container's server
- It should implement functions that allow steering(by AppContainer)
 - Setting parameters
 - Getting parameters
 - Controlling state (starting, stopping)
 - Errors throwing

The application plugin can be a java Japplet class if this is considered reasonable by its author, but it should define all necessary functions.

Generally there will be two kinds of such application plugins:

- Batch work application plugin
 - Application specific parameters setting (1st panel)
 - Pre-verification of parameters (1st panel)
 - Graphical parameters setting (eg. operations on land maps, points of which?are input for application) (1st panel)
 - Graphical visualisation of results if required (2nd panel)
- Interactive jobs application plugin
 - More complicated kind of plugin
 - Will contain all batch work plugins functionality
 - Will show interactive stream of joboutput (2nd panel)

8.2.2. Preparation and execution of the Application Plugin

1. The Application Container class will have information (URL, JAR file name etc.) how to get to the application plugin through the Internet. The application plugin developers should provide such information. For example URL like `http://ras.man.poznan.pl/gridApplication.jar` will allow the Application Container to download the application plugin class from this web page, to create an appropriate object and to start its execution inside the Application Container. The JAR file pointed by that URL should contain all the classes, resources and libraries required (imported) by the application plugin.
2. After the application plugin class is downloaded, a plugin object is created and initialised inside the Application Container. The next task of the Application Container is to add the application plugin to the proper GUI container (where the plugin will be displaying) and to prepare a separate thread for the execution of job output visualisation.
3. The application plugin must implement several methods defined in the `ApplicationPluginInterface`
 1. `void setInterfaceObject(ApplicationContainerAPI object);` This method allows the application plugin to get the reference to the Application Container API object (sending as a parameter of this method) that can be used to access some methods and objects provided by the Application Container in the near future. The `ApplicationContainerAPI` object allows having an interaction with specific methods and objects of Application Container from inside the application plugin.
 2. `JPanel getArgumentsPanel();` This method should return instance of `JPanel` which will be shown in `JobSubmissionWizzard` (as the "Arguments" tab). This panel allows to set-up arguments for the application.
 3. `JPanel getOutputPanel();` This method should return instance of `JPanel` which will be shown in `JobSubmissionWizzard` (as the "Job Output" tab). This panel allows visualising job output, and for interactive applications it can show on-line visualisation of application.
 4. `Hashtable getParameter();` This method should return a `Hashtable` of parameter - a set of pairs: `String->Object`; The "key" of this `Hashtable` contains the name of the parameter and the "value" contains the value of the related parameter. Possible parameters can be, for example, an array of arguments for application (e.g. `"-f data.in"`, `"-b=8"`, `"--verbose"`) or a set of suggested resource requirements.
 5. `void setParameters(Hashtable parameters);` This method allows to setup a set of parameters for the application plugin. Currently we expect to have at least one parameter – array of argument defined for a job. This array is the same as returned by `getArguments()` method and can be used to fill the arguments panel with proper values.
 6. `void go()` throws `ApplicationAppletException`; This method will be called to start data visualisation. The "go" method will throw a set of `ApplicationAppletExceptions`, which will be caught and maintained by the Application Container (in a separate thread run for that plugin). Such a set of special Exceptions must be implemented and provided by the application plugin developers.
 7. `void stop()`; This method informs the application plugin that the `JobOutputPanel` is not visible and visualisation can be stopped (e.g. visualisation thread can be suspended)

8.void start() ; This method informs the application plugin that the JobOutputPanel is now visible and visualisation can be started (e.g. visualisation thread can be resumed)

The ApplicationContainerAPI class is ready to provide an interface to communicate with the Application Container in case of necessity. Currently we cannot see any potential method necessary for plugin (feedback from applications strongly recommended).

Simple communication to the Application Container (e.g. suggest resource requirements or environment variables) can be done by setting/getting parameters.