



DELIVERABLE D4.4

TESTBED PROTOTYPE RELEASE

WORKPACKAGE 4

Document Filename:	CG4-D4.4-v1.2-CSIC10-WP4-PrototypeRelease.doc
Work package:	WP4 International Testbed Organisation
Partner(s):	CYFRONET, ICM, INS, UvA, II SAS, FZK, PSNC, UCY, TCD, CSIC, UAB, USC, DEMO, AUTH, LIP
Lead Partner:	CSIC
Config ID:	CG4-D4.4-v1.2-CSIC010-WP4-PrototypeRelease
Document classification:	PUBLIC

Abstract: This document reports on the testbed prototype release.



Delivery Slip

	Name	Partner	Date	Signature
From	Rafael Marco	CSIC	08-Jan-2002	
Verified by				
Approved by	Jesus Marco	CSIC	27-Jan-2002	

Document Log

Version	Date	Summary of changes	Author
1.0-DRAFT_A	8-Jan-02	Initial draft	Jesus Marco, Rafael Marco
1.1-DRAFT_B	27-Jan-02	Detailed draft including info from WP1, WP2 and WP3 deliverables	Jesus Marco, Rafael Marco
1.2-FINAL	18-Feb-02	Received comments from reviewers Integration plans confirmed	Jesus Marco, Rafael Marco

CONTENTS

1. INTRODUCTION	4
1.1. OVERVIEW.....	4
1.2. DESCRIPTION OF WORK.....	4
1.3. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS.....	5
1.4. REFERENCES.....	5
2. GENERAL OVERVIEW OF THE CROSSGRID TESTBED	6
2.1. THE CURRENT TESTBED STATUS: FIRST PROTOTYPE RELEASE.....	6
2.2. SUPPORTING APPLICATIONS AND MIDDLEWARE: DEVELOPMENT TESTBED.....	7
2.3. AN EXAMPLE OF TESTBED INTEGRATION INVOLVING ALL WORKPACKAGES: INSTALLING AND TESTING A SIMPLE MPI APPLICATION.....	8
2.3.1. <i>Description of the simple application example</i>	8
2.3.2. <i>Installation of the software at the FZK repository, and download procedure</i>	8
2.3.3. <i>Incorporating basic middleware: MPI with Globus</i>	9
2.3.4. <i>Running the application at a single testbed node</i>	9
2.3.5. <i>Running the application in the several nodes of the testbed site</i>	9
2.3.6. <i>Running the application with MPICH-G2 in the testbed site</i>	9
2.3.7. <i>Running the application across multiple sites</i>	10
2.3.8. <i>Incorporating the use of WP2 tools</i>	11
2.3.9. <i>Incorporating WP3 middleware</i>	11
3. REVISED RELEASE PROCEDURES	13
3.1. DEFINITION OF A “TESTBED RELEASE”.....	13
3.2. FORMAL STEPS FOR PREPARATION OF A NEW TESTBED RELEASE.....	14
3.2.1. <i>New software: installing at the CVS repository at FZK</i>	14
3.2.2. <i>Preparing the release proposal in the development testbed</i>	14
3.2.3. <i>Test and validation</i>	14
3.2.4. <i>Defining the testbed release components and calendar</i>	14
4. DEPLOYABLE SOFTWARE	15
4.1. THE BASIC TESTBED SOFTWARE: EDG MIDDLEWARE.....	15
4.2. EXTERNAL PACKAGES.....	15
4.3. WP2 TOOLS SOFTWARE.....	15
4.4. WP3 MIDDLEWARE.....	16
4.5. APPLICATIONS.....	16
5. INSTALLATION KIT AND DOCUMENTATION WEB SITE	18
6. UPDATE OF SOFTWARE REPOSITORY AND HELPDESK	19
6.1. SOFTWARE REPOSITORY USE.....	19
7. APPENDIXES	21
7.1. APPENDIX A: TESTBED EXTENSION AND SITE STATUS.....	21

1. INTRODUCTION

1.1. OVERVIEW

The first testbed prototype release has taken place in month 10, following the planning for testbed setup described in our first deliverable D4.1 (CG-4-D4.1-001-PLAN-1.1).

By month 9, as indicated in our previous deliverable D4.3, production testbed sites were running EDG middleware testbed version 1.2.2/3.

One of the main objectives of this document, deliverable D4.4, describing the testbed prototype release, is to provide all CrossGrid software developers (working on tools from WP2, middleware in WP3 and applications in WP1), with a description of the testbed possibilities, as their first prototypes will be released by M12 and subsequently tested.

The previous deliverable D4.3 provided a detailed testbed status report; this deliverable updates this status report in Appendix A, and concentrates more on the procedures for integrating the CrossGrid software, including a brief overview of the possibilities based on the current understanding of testbed people in contact with software developers, and also the limited experience from an MPI Example Application installed in the testbed.

The scheme of the document is as follows:

Section 2 provides a short general overview of the current testbed and its possibilities; the possibilities for support of application and middleware development in the development testbed are then described, followed by an MPI application example. Section 3 details the “release procedure”, covering from the basic middleware up to the application software release. Section 4 describes the deployable software, including a brief description of the requirements expected in the coming months from all workpackages, and a limited recount of the current experience and available manpower and computing resources. Section 5 describes formally the installation kit. Section 6 briefly updates the information about the use of the Grid Portal software repository at FZK, which was described in detail with the Help Desk in our previous deliverable. The current testbed status, including the resources description at each site, is given in Appendix A as indicated before. Also the status of Certification Authorities is included in section 6 of this appendix.

Further details of the integration process for applications, tools, and middleware, will be included as an Appendix in the coming Status Report corresponding to next deliverable (D4.5, due in M15). A first draft describing the progress already achieved thanks to the integration work in the CrossGrid meeting at Santiago (7-9th Feb 2003), resulting in the corresponding demo, is in preparation (see CG4-D4.5-v1.0-CSIC012-Integration.doc).

1.2. DESCRIPTION OF WORK

From Annex I: the product is D4.4 (Prototype) Month 10: First testbed prototype release (CSIC)

- Revised release procedures

- Deployable software including available packages at selected testbed sites

- Installation kit and documentation web site update

- Testbed evolution: from initial setup at selected sites to all testbed sites

- Setup of national CA, RA and LDAP servers and monitoring system.

- Update of Software Repository and Help Desk

1.3. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

CrossGrid/X#	The EU CrossGrid Project IST-2001-32243
DataGrid/EDG	The EU DataGrid Project IST-2000-25182
GRID	Grid framework for sharing of distributed resources.
HSM	Hierarchical Storage Manager
MPI	Message Passing Interface.
MPICH	A portable implementation of MPI (CH stands for Chamaleon)
WP	Work Package
VO	Virtual Organization
CE	Computing Element (EDG).
SE	Storage Element (EDG).
API	Application Programming Interface
MySQL	An Open Source Database following SQL92
RPM	Red Hat Package Manager
LCFG	Local Configuration Tool
OGSA	Open Grid Service Architecture

1.4. REFERENCES

[1] CrossGrid WP4 Deliverables

D4.1 (<http://www.eu-crossgrid.org/M3deliverables.htm>)

CG-4-D4.1-001-PLAN.pdf

CG-4-D4.1-002-SITES.pdf

CG-4-D4.1-003-SUPPORT.pdf

CG-4-D4.1-004-TEST.pdf

CG-4-PLAN-002-PROJECT.mpp

D4.2: (<http://www.eu-crossgrid.org/M6deliverables.htm>)

CG4-D4.2-v1.0-CSIC010-SetupOfFirstTestbed.pdf

CG4.4-D4.2-v1.1-LIP011-ValidationOfTestbedArchitecture.pdf

D4.3: (<http://www.eu-crossgrid.org/M9deliverables.htm>)

CG4-D4.3-v1.1-CSIC010-WP4-Status.pdf

CG-4.4-D4.3-v1.3-LIP012-InitialCrossGridTestbed.pdf

2. GENERAL OVERVIEW OF THE CROSSGRID TESTBED

2.1. THE CURRENT TESTBED STATUS: FIRST PROTOTYPE RELEASE

The objective of the testbed in the CrossGrid project is to provide a framework to test and run the applications and corresponding middleware in a realistic distributed environment.

The CrossGrid project stated as one of its main objectives the extension of the Grid to new regions and countries in Europe, assuring interoperability with the framework provided by the European DataGrid project (EDG), based on the Globus middleware. To achieve this objective, the CrossGrid testbed has been developed in contact with the EDG work teams, and the basic middleware deployed corresponds to the EDG testbed software versions 1.2.x and 1.4.x, supporting Intel IA32 architecture on Linux Red Hat v 6.2. Also a common policy on certification has been agreed, and user support tools like the Software repository and the Helpdesk are evolving in close contact.

The first testbed prototype release is the result of the work along these lines during the first 10 months of the project. It includes, as planned, two separate sets of distributed computing resources:

- The **Production testbed** extends over all sites (LIP, CYFRONET, ICM, INS, UvA, IISAS, FZK, PSNC, UCY, TCD, CSIC, UAB, USC, DEMO, AUTH). It provides the basic framework for running applications showing the Grid potential. The current production testbed is based in the EDG release 1.2.2 . The production testbed is by definition stable, and new releases of the basic Globus and EDG software will be incorporated only after a complete test and validation process. The same procedure applies to the CrossGrid application and middleware prototypes.
- The **Test and Validation testbed** supports this effort on a reduced number of sites (LIP, CSIC, FZK, DEMO). The current version is based in the EDG release 1.4.x .

Appendix A describes in great detail the status, usage, and experience regarding the CrossGrid production and test and validation testbeds, including site resources and possible testbed extensions.

Any complete testbed includes distributed resources available at each site plus a set of central services. The minimum resource list at each site includes several machines:

- a **Gatekeeper**, providing access to local **Working Nodes** through the local batch scheduling system
- an **Storage Element** to interface any storage resource (from disk pools to HSM systems)
- a **User Interface** machine used to submit jobs to the testbed.
- The **LCFG installation server** is used to configure and maintain all the above systems.

The central services, located at LIP, for the production testbed include a resource broker machine plus an authentication credential proxy, a replica catalog that provides the location of physical files in the grid, a virtual organization server hosting the list CrossGrid of users, and the network-monitoring platform, mapcenter, that provides the view of the production testbed status displayed below

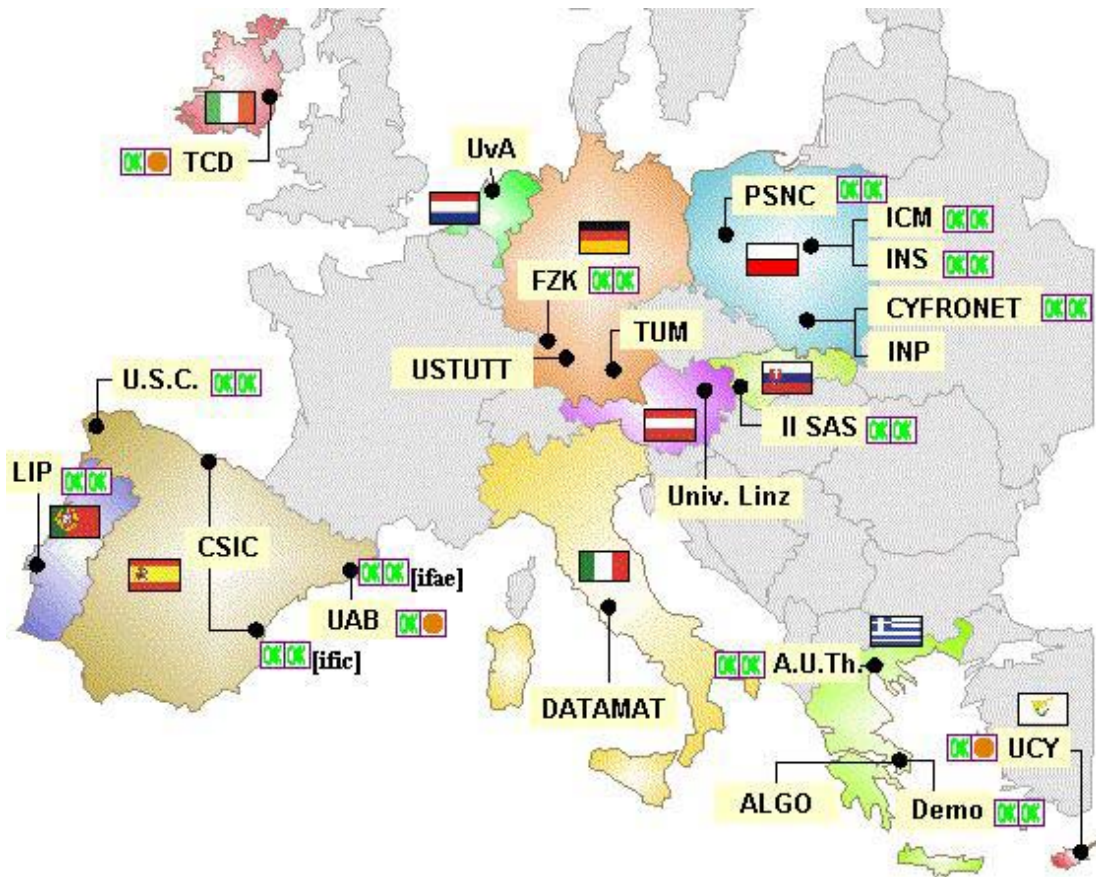


Figure 1: Mapcenter showing a geographical view of the CrossGrid Production Testbed.

2.2. SUPPORTING APPLICATIONS AND MIDDLEWARE: DEVELOPMENT TESTBED

The current production testbed fulfills the objective of extending the Grid to new regions and countries in Europe, assuring interoperability with DataGrid.

However the CrossGrid testbed aims to support interactive distributed applications, and also the development of these applications and new supporting tools and middleware. An initial study of the requirements stated by the corresponding CrossGrid workpackages (WP1, WP2 WP3) and the Architecture Team (TAT) was presented in our first deliverable D4.1, based on the information provided in the Software Requirements Specifications (SRS) documents. Many requirements are not currently explicitly supported by the EDG middleware distribution, like for example the support for parallel distributed applications based on MPI. Others do imply a significant change in existing services, like the work on scheduling agents for interactive applications. Some even imply, during the development stage, small but relevant changes to the operating system kernel, like accessing hardware counters for monitoring. And finally others require the deployment of extra resources in the testbed, like web servers for application portals. The **Development testbed** will provide the resources needed for this software development in a grid aware framework. Minimal but flexible configurations will be deployed at different sites to address different workpackage development tasks: UvA, IISAS and CSIC for WP1, FZK, UCY, AUTH for WP2, PSNC, CSIC, UAB, TCD, CYFRONET for WP3.

With the first application and middleware prototypes being deployed along the coming months, it is quite important to provide to the developers with a clear description of what they can expect from a development testbed, and the path for integration in the production testbed, after the test and validation procedure.

We have preferred to start with a real example that corresponds to a simple MPI application. Note that this is a real example, with real code posted in the FZK repository in December, and tests done first in local clusters (taking partially the role of an initial development testbed), and then in the test and validation testbed and the production testbed, but without a new testbed software release.

In the more complex plans for deployable software for all workpackages (see section 4 below) after the work in the development testbeds, the software will be submitted to the WP4 for integration. The WP4 integration team will take care of proposing the new software to the test and validation testbed. If this step is successful, it will be incorporated in the next testbed prototype release, and propagated to the production testbed.

2.3. AN EXAMPLE OF TESTBED INTEGRATION INVOLVING ALL WORKPACKAGES: INSTALLING AND TESTING A SIMPLE MPI APPLICATION

This section tries to introduce a general perspective on the use of the testbed for developers, testbed integrators and site administrators with the help of a simple MPI application: the ExampleApplication developed in Task 1.3 for test purposes. It is not intended as a formal guide, for more details see below the preliminary discussion on testbed integration possibilities, and section 5 in Appendix A for more details on the testbed usage along the last months.

2.3.1. Description of the simple application example

The ExampleApplication is a simple program written in C, that has been used as a basic test on how to run MPI applications in the CrossGrid testbed.

The program includes a master process plus several slaves; the master triggers the generation in each slave of random numbers with a uniform distribution and a given mean, and collects the mean obtained in each slave to obtain a global mean. Input is simply the given mean and the resources (number of nodes) to be used; the output is the obtained mean. The program also prints out the partial mean obtained by each slave node. It was written and tested in a normal desktop machine running RH6.2, with the MPI package installed.

Despite its simplicity, the scheme used is similar to other more complicated applications like the distributed training of a neural network for HEP in task 1.3 from whom it was derived. In particular the use of MPI calls is hidden under more generic calls so any evolution to a new framework for distributed computing will need only to replace these routines, named “Common_DistLib”.

We are considering a similar complete description of this integration chain using the HEP distributed neural network training prototype, regarding the testbed demonstration for the annual review.

2.3.2. Installation of the software at the FZK repository, and download procedure

The ExampleApplication software was posted at the CVS repository at FZK; it can be found at http://gridportal.fzk.de/cgi-bin/viewcvs.cgi/crossgrid/crossgrid/wp1/wp1_3-hep/src/example/ where the source code and scripts to test it are available

To download the application and install it, either the CVS commands can be used, or directly the available “tarball”. The executable for the current testbed machines (IA32 with RH6.2) is also included.

2.3.3. Incorporating basic middleware: MPI with Globus

The MPI package MPICH-G2 is needed first of all for compiling the MPI application for use in the Globus environment. At running time the package is not needed if the application is statically linked; this is the case in this first example.

An MPI package like MPICH-G2 should be incorporated in the testbed software as an external package for general use. EDG will incorporate MPI likely in the future release version 2.0 around May 2003. In CrossGrid, as application, tools and middleware prototypes will use MPI, we will need to include it in the next testbed release. As any middleware, before being incorporated to the production testbed it has to go through the test and validation procedure described in Appendix D of our first deliverable D4.1. This process has started, and moreover the positive experience running MPI in the testbed has already been reported in our previous deliverable, and is described with more detail in Appendix A of this deliverable. However the formal request to the WP4 Integration team for test and validation of this middleware has to come after the experience in a development testbed.

In principle MPI can be incorporated as a basic package to all testbed nodes, as it is relatively lightweight, and useful if in the future the application is dynamically linked. By now we assume that it is incorporated at least in a machine at the testbed considered for compilation purpose. Among the different testbed elements, and after looking to their different functionality, the User Interface machine seems to be the most obvious initial candidate: a machine allowing interactive login of a developer where the compilation can take place. In resume, we propose to include in the development testbed a UI machine with the MPICH-G2 package installed. Note that this is also a non invasive basic middleware, as it doesn't interfere with the current UI software. However it requires administrator privileges for the installation. We also propose to install an MPI package not requiring Globus, for reference purposes (see below). It also helps to understand the source of possible problems (i.e., if the application runs with MPI but not with MPICH-G2, etc.)

2.3.4. Running the application at a single testbed node

To compile and then run the ExampleApplication at the development testbed site, the developer needs first an account in this UI machine. Note that such accounts are provided locally, by the site administrator. The UI needs to have MPI installed (see above). Once logged in the UI in a local account, you can download the ExampleApplication tarball from the CVS repository, and run the example in a single node with the mpi-run command, specifying only one machine (the UI machine) in the pgfile describing the resources, acting as master and slave processes. The detailed recipe can be found in the repository. Both MPI and MPICH-G2 (see below) versions should be tested.

2.3.5. Running the application in the several nodes of the testbed site

For this step, at least two machines are needed in the development testbed. The mpi ExampleApplication can be tested now with two or more slaves by specifying these machines in the pgfile. Take the provided pgfile and modify it to include the machines in the testbed, identifying with a 0 the machine where the master process will run, and with a 1 the nodes where slaves will run; the simplest way for the application distribution, is to copy the executable file to your account on those machines. This is a mandatory reference step for future comparison before running the application using a Globus mpi context.

2.3.6. Running the application with MPICH-G2 in the testbed site

Now the user needs to install his/her certificate in the UI machine, recompile the application using MPICH-G2, and launch it with the globus-run command indicating the list of resources in the RSL file.

To use the certificates, each user should contact the national CA authority, and then copy the public key certificates (files usercert.pem and userkey.pem) to the usual directory .globus and set the

adequate file permission (444 and 400 respectively). The command `grid-proxy-init` creates then the proxy certificate to enter the testbed. Host certificates are also needed (`hostcert.pem` and `hostkey.pem` should be in the `/etc/grid-security` directory).

Take note that the machine also needs a public port open, 2119 in our case, that will be used for the `gsi` gatekeeper service, so it has to be added in `/etc/xinetd.d`. These last two steps, as well as the addition of users in the `/etc/grid-security/grid-mapfile` do require direct administrator intervention, and should be included in the development testbed setup.

Explicitly, to prepare the ExampleApplication to run in this environment, recompilation with MPICH-G2 is needed: edit the Makefile to redefine

```
> MPICC=/opt/mpich/bin/mpicc (or the adequate path)
```

and rebuild using the `make` command. Now the application can be run with the following command:

```
>/opt/mpich/bin/mpirun -globusrsl test.rsl
```

where `test.rsl`, that uses the resource specification language of Globus, is the script to run the program.

As an example we add here the `test.rsl` file used for the test at IFCA site:

```
( &(resourceManagerContact="grid066.ifca.unican.es")
(count=1)
(label="ANN 0")
(environment=(GLOBUS_DUROC_SUBJOB_INDEX 0)
(LD_LIBRARY_PATH /opt/globus/lib/))
(directory=/afs/ifca.unican.es/user/mrivero/mpi/examples)
(executable=Example_opt.exe)
(stdout=/afs/ifca.unican.es/user/mrivero/mpi/examples/test.txt)
)

( &(resourceManagerContact="grid067.ifca.unican.es")
(count=1)
(label="ANN 1")
(environment=(GLOBUS_DUROC_SUBJOB_INDEX 1)
(LD_LIBRARY_PATH /opt/globus/lib/))
(directory=/afs/ifca.unican.es/user/mrivero/mpi/examples)
(executable=Example_opt.exe)
(stdout=/afs/ifca.unican.es/user/mrivero/mpi/examples/test.txt)
)
```

See Appendix A, section 5 for more details and possibilities regarding this step.

2.3.7. Running the application across multiple sites

This is the latest and most complicated step. While waiting for the adequate interactive scheduler and resource broker, a manual procedure for running as the one described above could be extended including multiple sites. The user needs that his/her certificates can be accepted at other sites (assured inside the CrossGrid testbed) and that she/he has authorization to use the resources (given by the VO). So the user should be included in the corresponding VO (in this case the simplest possibility is to use the "crossgrid" VO). The matchmaking between generic accounts and VO user accounts eliminates the need to have local accounts in all testbed machines for registered users, when running the application. Note however that this will require the setup of the development testbed resources for this scheme.

2.3.8. Incorporating the use of WP2 tools

WP2 provides (see D2.3) new tools for the development of grid applications, like the MPI code debugger/profiling interface developed in task 2.2, the grid benchmarking tool from task 2.3, or the grid performance prediction tool from task 2.4. To improve applications developed in WP1, the use of this tools is suggested, so they should be installed and available in the testbed. We describe here briefly the case for the code of the MPI code debugger/profiling interface MARMOT (see D2.3 section II).

The installation of tools should take into account the testbed functionalities and the tool installation requirements on software and at running time. These issues require a discussion in common between the WP2 and WP4 integration teams. The first proposal is to adapt the User Interface machine to allow both the WP2 developer and the WP1 developer the compilation of MPI applications in this machine, providing a Developer Workstation service.

This requires the definition of a new LCFG profile, Developer Workstation (DW), incorporating the RPMs corresponding to the MARMOT package, scripts, etc. The first step requires downloading the code from the Grid Portal CVS repository for task 2.2, and the creation of the corresponding RPMs, that will be incorporated to the test and validation testbed release software repository under the CrossGrid software area. The new DW profile itself should be as well incorporated to the repository.

The developers with the help of the Integration team should select an available UI machine in the development testbed, or install a new one, and try to upgrade it into a Developer Workstation.

After successful installation of this machine, the WP2 developer can login and verify that the tool performs as expected employing the corresponding software tests. The tool software has to be submitted then to the WP4 integration team. The WP1 developer can now try the tool on his own application using the machine installed in the development testbed with the WP2 tools, but it will be likely more efficient to wait for the validation of the tool, and the corresponding upgrade in the development testbed.

A similar scheme could be possible for incorporating other WP2 tools: the performance prediction, grid benchmarking and the grid monitoring tools.

2.3.9. Incorporating WP3 middleware

CrossGrid WP3 is developing middleware (see D3.3) to be used in the context of Grid interactive applications. The WP3 integration team has provided a detailed installation guide, and common discussion between WP3 and WP4 integration teams is being done to address a development testbed deployment.

The developed middleware includes (see below for details) a roaming access server and an application portal server (task 3.1), the modified scheduler (task 3.2), monitoring middleware (task 3.3) for network tracing (SANTA-G), resources monitoring (OCM-G), and Jiro monitoring. And finally the software for Data Access Optimization (task 3.4)

Some of these packages require a substantial modification of the testbed elements, at the system administrator level, or the introduction of new ones. The initial proposal here presented tries, redefining services, to minimize the impact on the current resources and the need for new ones.

First it has to be noted that the UI will be modified for the new scheduler (task 3.2) as well as RB and II machines. The integration of a user browser (Netscape for example) in the 'old' UI (note that it was previously "updated" to Developer Workstation, and this makes sense) could make it useful also for accessing the Application Portal and the Data Optimization service.

For the Application Portal Server and the Roaming Access Server, the initial proposal is to deploy both linked to the corresponding Virtual Organization. This makes sense regarding ubiquitous

access to user resources, including data files and metadata, profiles, scripts, etc. A new node, (Application & Portal Server) would incorporate these services.

For the ExampleApplication the test checks only the inclusion of the Application Portal Server: an Apache server is installed in a new machine, and also the MySQL and php packages. A simple script providing XML input and output through it has to be implemented, and the communication proceeds then directly to a service running in the old UI (now the DW) both for user output/input and for submission to testbed resources.

Once the machine is installed, configured and running, the integration in testbed releases should proceed as for the WP2 tools.

3. REVISED RELEASE PROCEDURES

A new testbed prototype release takes place when a significant amount of software incorporating new modules or relevant changes with respect to previous versions is considered to be ready for deployment in the production testbed.

As planned when preparing the project, the procedures for software release have been revised according to the experience in the first months of project life. The “*testbed incremental release*” procedure proposed in the “Detailed planning and for testbed setup” document D4.1, indicated that “*maintaining as much as possible the working infrastructure, new improvements and developments will be added in an as soft and stepped way as possible*”.

The experience with the CrossGrid testbed setup, described in the deliverables D4.2 and D4.3, has shown that it is important to have a clear release procedure to optimize the testbed management. The frequent EDG releases (from version 1.1 to current 1.4) have not been followed immediately by all CrossGrid sites: the production testbed including most sites has deployed EDG versions 1.2.2 / 1.2.3, while only the sites involved in test and validation have followed EDG up to version 1.4.4. This situation corresponds to this first testbed prototype release.

The situation may change substantially in the coming months with the first CrossGrid application and middleware prototypes, and requires a more detailed specification of the release procedure.

3.1. DEFINITION OF A “TESTBED RELEASE”

A “testbed release” identifies a well defined and complete collection of software to be deployed on a well defined set of distributed computing resources, including the corresponding installation mechanisms.

According to the CrossGrid Architecture (see D5.2.2), the testbed will support the software components which will be developed within the framework of the project (CG software) and those from Globus and DataGrid (collectively referred as EDG software).

EDG release packages (see EDG D6.4) are designed to work with the LCFG installation tool. They are available from their official CVS repository. A similar scheme will be used in CrossGrid. The release includes the software and corresponding configuration to be used in each testbed resource: LCFG templates include the list of packages (RPMs for the RedHat environment), a configuration file that has to be customized at each site, with the corresponding instructions.

For convenience reasons, the full CrossGrid testbed release, including the EDG packages, is available from a single entry point at the code repository GridPortal hosted by FZK:

<http://gridportal.fzk.de/distribution/crossgrid/releases/>

The current production and test and validation testbed packages can be found there, and the development testbeds will be hosted also there when required.

Below the previous link, three different areas can be found for each release:

- The **cg** area includes the Crossgrid software
- The **edg** area includes the copy of the corresponding EDG release
- The **other** directory hosts the external software not included in the EDG release

The current production testbed version has been labeled as 0.0.1. It corresponds to the EDG version 1.2.3. The current test and validation testbed version is 0.0.3, and corresponds to EDG version 1.4.2. The **edg** subdirectory is labeled after the EDG version numbering to avoid misunderstanding (i.e., edg-v1_4_2), but is unique for each version.

More detailed instructions for testbed deployment can be found in Appendix A, and a step by step installation guide complementing those of EDG (see the EDG WP6 documentation) is been prepared adapted to the use of this distribution download area. With the LCFG tool, these are the essential components of the CrossGrid installation kit (see below).

3.2. FORMAL STEPS FOR PREPARATION OF A NEW TESTBED RELEASE

The preparation of a new testbed release is triggered by the need to incorporate new software either developed inside the project or externally.

We consider here testbed releases aimed to reach the production testbed: releases in the development testbed can be as frequent as needed as long as the resources are available, while the test and validation testbed procedures has been described in detail in our first deliverable D4.1

The testbed release requires the agreement of the different WP integration teams, and in particular of the WP4 integration team. Requests should be addressed in principle through the WP4 coordinator. Typical requests include the need to deploy new applications or middleware, keep EDG compatibility, fix serious bugs, etc.

3.2.1. New software: installing at the CVS repository at FZK

The first mandatory step for any new software to be deployed in the future in the CrossGrid testbed is its installation in the CVS repository hosted by the Grid Portal at FZK. Details about its functionalities and use can be seen in the previous deliverable, D4.3. The software in the corresponding WP tasks directories will be copied to the distribution area when the release is ready.

External software not directly related to an application, tool or middleware, will be posted in the WP4 area. In particular the EDG software is copied by the WP 4.2 (Coordination with DataGrid) task people at FZK for each new EDG release.

3.2.2. Preparing the release proposal in the development testbed

The development testbed should download the new or modified software from the corresponding CVS repositories, and install it. Involving developers and, in what possible, WP4 people in this step should guarantee that modifications are incorporated to the software repository and also that RPMs are prepared for the software components, and also the corresponding configuration files for LCFG. The software with corresponding tests, and the documentation, should be posted and available in the corresponding software repository.

3.2.3. Test and validation

As stated in the middleware test procedure in D4.1, developers should transmit to the WP4 integration team new or updated middleware components, but also the WP4 integration team should identify them. This is in particular true for external packages. The test procedure itself is carried mainly by the WP4 task 4.4 in the test and validation testbed, and includes two different phases: in the alpha phase each delivered component is tested first “alone” and then integrated with other components and across test sites; in the beta phase knowledgeable users and other middleware developers are invited to perform tests using real applications.

3.2.4. Defining the testbed release components and calendar

Reaching a complete and coherent set of software components, tested and validated, that can be deployed in the production testbed is under the responsibility of the WP4 integration team.

Too frequent releases of the production testbed are not desirable. Usually they will take place in coordination with major software releases in WP1, WP2 or WP3. So frequency will be in the range of 3-6 months. Releases in the test and validation testbed will be much more frequent.

4. DEPLOYABLE SOFTWARE

This section tries to provide a brief but indicative path for CrossGrid software deployment in next testbed releases, and has been prepared with the help of WP4 members in close contact with WP1, WP2 and WP3 work packages.

4.1. THE BASIC TESTBED SOFTWARE: EDG MIDDLEWARE

As indicated, EDG middleware is obtained from the EDG CVS repository, and included in the CrossGrid CVS repository by the FZK team responsible for task 4.2 (Coordination with DataGrid). Test and validation effort on this software is carried systematically by LIP and DEMO under task 4.4.

EDG middleware version 1.2.2/3 is used in the production testbed. The current EDG middleware version 1.4, installed in the test and development testbed, has been found to be stable and providing significant improvements. It is expected that the next CrossGrid production testbed release will use this version, as it will likely take place before the next major EDG middleware release (version 2.0).

Some of the components of the EDG middleware provide also the basis for new CrossGrid middleware. As an example, the new scheduler scheme for MPI parallel applications is based in the EDG WP1 software.

4.2. EXTERNAL PACKAGES

Several external packages will be required to support CrossGrid applications, tools and middleware. An example is the MPICH-G2 package cited above. Other examples include web server modules, latest java packages, XML parsers, etc. A coordination effort to minimize the number of these packages and assemble a coherent set will be done by the WP4 integration team.

One of the main points is to reuse when applicable the external software included in the EDG release: for example advice has been provided to the developers in task 1.3 to use the *libwww* module or the *XML parser* modules included in the EDG distribution. This effort is being extended to all tasks as the first prototype descriptions are released.

4.3. WP2 TOOLS SOFTWARE

As indicated, WP2 tools are being already considered for integration in the development testbed. The common scheme agreed between the WP2 and WP4 Integration teams is to evolve a User Interface into a Developer Workstation, and install the tools software in this machine to be used by the application developers. The first steps along this direction are being prepared:

The MPI verification tool, MARMOT, will be included in a development testbed supported by CSIC-IFCA in the WP4 integration team. The MPICH-G2 package is the main external software to be deployed. This testbed will support also the integration of the application software developed in tasks 1.3 and 1.4.2.

The Grid benchmarking tool will be supported in a development testbed by the main developer, UCY, and by AUTH member of the WP4 integration team. It requires the installation of a Linear Algebra package.

The Prediction Performance tool developed by USC will be integrated with the help of CESGA people in WP4 supporting the site testbed.

The OCM-G monitoring tool is proposed to be tested at CYFRONET. It requires the use of the X11 graphical interface, supported locally.

The demonstration and report on these first prototypes of all the WP2 tools is available now including test scenarios, evaluation suite, and the documentation of design, implementation and interfaces. The software itself is being posted to the FZK Grid Portal repository.

4.4. WP3 MIDDLEWARE

Similarly to WP2, the demonstration and report of the first WP3 prototypes is already available, and moreover the software itself is already posted in the corresponding area at the FZK Grid Portal repository in the case of most tasks.

The path to integration in the testbed established in agreement between the WP4 and WP3 integration teams includes the following deployments to the development testbed:

For the portal deployment, task 3.1, common work between developers at PSNC, application users at CSIC-IFCA and WP4 development testbed supporters at both sites will take place. A new Application Server machine will be installed to provide the support for the Portal Application Server, the Migrating Desktop, and the Roaming Access Server. The machine will be included in the corresponding VO (Crossgrid by now), and managed as such regarding user access, etc.

Work on the new scheduler being developed in task 3.2 requires the modification among other elements of the Resource Broker. The new software is being deployed at the CSIC-IFIC site, and will be also tested at UAB.

OCM-G monitoring middleware will be tested at CYFRONET. In the current situation, it requires small but significant patches to the current Linux kernel. The first integration work will be a workaround to the current patch for *pipe.c* enabling *FIFOs*. A more ambitious attempt to solve the problem related to use of hardware counters, may require its inclusion in future Linux kernels or RH distributions.

The SANTA-G monitoring will be incorporated to the testbed with the help of TCD. It requires installation of the RGMA service, which is included in the latest EDG release.

The Data Access Optimization middleware proposes a deployment framework. It mainly touches the Storage Element, in a non- “destructive” mode. It is being tested inside WP3 at CSIC-IFIC and the development testbed will include likely resources at CYFRONET. In particular this could be the only possibility to access significant resources like a Hierarchical Storage Manager.

No plans are available yet for integrating Jiro monitoring in the testbed.

Finally ICM will take care of the deployment into the testbed of the application for post-processing of monitoring data. It has already been run in a testbed Worker Node, but the main challenge is to test it on the output of the SANTA-G monitoring tool, and provide a clear path for integration and use by application developers.

4.5. APPLICATIONS

First internal software releases running in local clusters start to be available now, and will require integration work for future testbed deployment.

Task 1.2, Flood Crisis Team Support, already includes in its first internal software a path for testbed integration for meteorological, hydrological and hydraulic simulations. The meteorological simulation requires MPI, and shares with task 1.3 and 1.4 similar requirements on its use, that will have to be fixed in common mode until the new CrossGrid Scheduler with parallel capabilities is available. The use of the Storage Element for data management is also proposed. The hydrological and hydraulic simulations are on the other hand high throughput applications, and the integration in the testbed should be straight forward. Finally the portal functionality should be offered in the framework proposed by the Application Portal Server. IISAS is providing the development testbed for this integration effort.

Task 1.3, Interactive Analysis of HEP data, will follow the deployment of the ExampleApplication described above, with the release of the distributed training Neural Network application. The most significant difference is the use of a data sample. A reduced sample has been posted at the download repository for initial test purposes. In the development testbed at CSIC-IFCA it

will be stored using a single disk server as storage element, or stripped across a disk pool close to the worker nodes. This work will be related to the first prototype for distributed database access being prepared, that will be common to task 1.4.2.

Tasks 1.1 and 1.4 do not have yet detailed plans for testbed integration. The Biomedical application is proposing the use of OGSA, not yet available in our testbed framework. Task 1.4, covering meteo applications, is presenting now the results of migration of data mining techniques to the Grid. In particular for task 1.4.2, aiming to mesoscale predictions, the Self Organizing Map algorithm has been started to be tested using MPI at a local testbed at CSIC-IFCA. Similarly the MPI application for Air Pollution will be tested by USC at the CESGA testbed.

5. INSTALLATION KIT AND DOCUMENTATION WEB SITE

The installation kit for the current testbed, as described above, is based on:

- The release software available from the single download link at the Grid Portal in FZK
- The corresponding installation documentation: the new EDG installation guides and the comprehensive CrossGrid installation guide.
- The LCFG package used as installation tool, including profiles and configuration files, and available inside the release software.

Appendix A includes in section 8, Testbed Deployment, further details about middleware installation and configuration (see sections 8.1 and 8.2).

Documentation is available from the Crossgrid WP4 web pages, see:

<http://grid.ifca.unican.es/crossgrid/wp4>

Specific information on the current testbed status is available following the corresponding link or directly from the LIP site:

<http://www.lip.pt/computing/cg-services>

6. UPDATE OF SOFTWARE REPOSITORY AND HELPDESK

The CrossGrid Software Repository and the HelpDesk were described in detail in our previous deliverable D4.3 (see sections 2.1 and 2.2). As most workpackages have started to post the software at the Grid Portal repository at FZK, a brief update showing the current (January 03) use of the software repository is given below. The Help Desk will get more user input and feedback as the developers start to participate in new testbed deployments, an activity starting now. The experience will be described in the next WP4 status report.

6.1. SOFTWARE REPOSITORY USE

As Grid Portal is organized in Projects, a project for each task has been created. Currently the projects are being explored by the users. I.e. users are getting accounts, request membership in their task's project and start uploading and using the CVS repository.

The following table shows a detailed summary of the current usage of the CVS repository.

Task		
Complete Project	Size:	169M
	Files:	3661
	Code lines:	590416
	Documentation:	12789
	.doc + .pdf files:	162
	rpm-files:	6
Wp1/wp1_3-hep	Size:	1.4M
	Files:	23
	Code lines:	10335
	Documentation:	67
	.doc + .pdf files:	0
	rpm-files:	0
wp3/wp3_1-portals	Size:	64M
	Files:	516
	Code lines:	214049
	Documentation:	34
	.doc + .pdf files:	26
	rpm-files:	6
wp3/wp3_2-scheduling	Size:	34M
	Files:	740
	Code lines:	95819
	Documentation:	9094
	.doc + .pdf files:	0
	rpm-files:	0
wp3/wp3_3-moninfr	Size:	6.2M
	Files:	533
	Code lines:	99599
	Documentation:	27
	.doc + .pdf files:	8

	rpm-files:	0
	Size:	5.4M
	Files:	533
	Code lines:	99599
wp3/wp3_3-moninfr/wp3_3_1-ocm-g	Documentation:	27
	.doc + .pdf files:	8
	rpm-files:	0
	Size:	50M
	Files:	1347
	Code lines:	128575
wp3/wp3_4-expert	Documentation:	2894
	.doc + .pdf files:	122
	rpm-files:	0
	Size:	5.9M
	Files:	4
	Code lines:	0
wp3/wp3_5-integration	Documentation:	0
	.doc + .pdf files:	6
	rpm-files:	0
	Size:	1.3M
	Files:	105
	Code lines:	9636
wp4/sites/demo	Documentation:	0
	.doc + .pdf files:	0
	rpm-files:	0
	Size:	1.4M
	Files:	186
	Code lines:	17101
wp4/sites/fzk	Documentation:	514
	.doc + .pdf files:	0
	rpm-files:	0
	Size:	756k
	Files:	97
	Code lines:	9580
Wp4/sites/lip	Documentation:	0
	.doc + .pdf files:	0
	rpm-files:	0
	Size:	672k
	Files:	77
	Code lines:	4980
Wp4/sites/psnc	Documentation:	0
	.doc + .pdf files:	0
	rpm-files:	0

7. APPENDIXES

7.1. APPENDIX A: TESTBED EXTENSION AND SITE STATUS

See attached document CG4.4-D4.4-v1.0-LIP013-CrossGridTestbed-final.doc

Abstract: This document describes the status of the CrossGrid *production* and *test and validation* testbeds. It also includes a brief report on the Certification Authorities (see section 6).