TEST AND INTEGRATION PROCESS DESCRIPTION

DELIVERABLE D.3.2

Task 3.5 Test and Integration

Document Filename:	**CG3.5-D3.2-v1.0-CSIC-TestIntegration.doc**
Work package:	**WP3 New Grid Services and Tools**
Partner(s):	**PSNC, CYFRONET, UAB, CSIC**
Lead Partner:	**CSIC**
Config ID:	**CG3.5-D3.2-v1.0-CSIC-Final**
Document classification:	**Public**

Abstract: This document provides a description of the WP3 test procedure and the status of the integration process in month 6, focusing on first milestone is in month 12.

Delivery Slip

	Name	Partner	Date	Signature
From	Task3.5, WP3	CSIC	30/7/2002	
Verified by	Marek Garbacz (X#TAT) Alfredo Tirado-Ramos (WP1)	CYFRONET UvA	24/8/2002	
Approved by				

Document Log

Version	Date	Summary of changes	Author
1-0-DRAFT-A	16/8/2002	Draft version	Santiago González de la Hoz, Lukasz Dutka, Miquel Senar, Miroslaw Kupczyk, Bartosz Balis, Brian Coghlan
CG3.5-D3.2-v1.0-CSIC020	28/8/2002	Final Version after internal review	Santiago González de la Hoz

CONTENTS

1. EXECUTIVE SUMMARY

Section 2 describes briefly a set of tools and services, which will be developed by the WP3 and will define the middleware layer of the CrossGrid project. Also a short description of the main goals and dependencies of Task 3.5 is provided. Section 3 details the References. Section 4 describes the test procedures for each module, which will be used during the test and integration phase by each Task. This process requires first local testbed set-up, available on selected sites. Section 5 contains information concerning workflow and interfaces to other workpackages. The interactions with other Tasks are described. Finally, section 6 provides a description of the integration process status in month 6 and explains how the tools and services developed in the WP3 will be integrated to produce an official version of the WP3 – middleware, how they will be integrated with Globus, DataGrid software and how the integration will be tested.

2. INTRODUCTION

2.1. PURPOSE

The main goals of the tests and integration are to deliver a release version of WP3 before each prototype [1]. WP3 will develop Grid services and a software infrastructure required to support the Grid users, applications and tools as defined in the workpackages WP1 [2] and WP2 [3]. This workpackage includes a set of tools and services, which will define (including the results of WP2) the middleware layer of the CrossGrid project.

There are four subject in this workpackage. The first is to establish a user friendly Grid environment by portal access to the Grid independently from the user location [4]. This Task includes two subtasks. The first one will allow access to applications via portals, which will support the users directly by simplifying the handling of applications. The second subtask will create a mechanism for allowing one to enter the Grid environment from any place and from any hardware (independently of the type of the operating system) with the same previously used user's environment. The second subject addresses the construction of new resource management techniques [5], namely the design and evaluation of self-adaptive scheduling agents that will be responsible for scheduling parallel applications submitted to a Grid. The goal is to achieve a reasonable trade-off between the efficiency of resource usage and application speedup according to the user's optimisation preferences. In the third subject we propose to develop a prototype infrastructure for the needs of monitoring-related activities for automatic extraction of high level performance properties and for tool support of performance analysis [6]. We also propose to build a tool for monitoring network infrastructure. Another aspect of the Grid performance will be solved in the fourth subject: in order to help users in their interaction with large data manipulation, we propose a kind of expert system which will operate as and advisor for the selection of migration/replication policy in a particular environment [7]. Within this subject a common middleware for fast tape file access and system for data access time estimation will also be developed.

Task 3.5 will prepare an official prototype version of the whole set of tools of the WP3 – middleware [8]. This middleware software will be used by other WPs, especially the WP1 and WP2. Task 3.5 will deliver all information about possible errors detected while the system tests and also about required corrections which should be done, including e.g. changes in the architecture or the API interfaces between the WP3 tools and other modules (WPs). The first prototype of WP3 software for local grids will be available in M 12, and a prototype for a global grid infrastructure will be available in M 24.

Task 3.5 is very strongly correlated with the prototypes released in WP4 [9]. The testbed supported by WP4 will be used in Task 3.5 as a testing environment, allowing testing separate tools (Tasks: 3.1 … 3.4) as well as the whole prototype of WP3 (middleware). Task 3.5 requires that the local grid testbed be available in M 6 (with a Certification Authority, Registration Authority, monitoring system and access to data storage system) and the full grid be available in M 12.

2.2. DEFINITIONS, ABBREVIATIONS, ACRONYMS

Definitions

Globus	The Globus Toolkit is a set of software tools and libraries aimed at the building of computational grids and grid-based applications
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.

Condor	Condor is a High Throughput Computing (HTC) environment that can manage very large collections of distributively owned workstations.
Condor-G	Condor-G is a resource manager that is used as a front-end to a computational Grid. It provides job monitoring, logging, notification, policy enforcement, fault tolerance and credential management.
GridFTP	It is a high-performance, secure, reliable data transfer protocol optimised for High-bandwidth wide-area networks.
CoG Kits	They allow the use of basic Grid services through commodity technologies such as framework, environments, and languages.

Acronyms/Glossary/Abbreviations

API	Application Programming Interface
CGRB	CrossGrid Resource Broker
CGSA	CrossGrid Scheduling Agent
CVS	Concurrent Versions System
FMA	Federated Management Architecture
GUI	Graphical User Interfaces
GSI	Grid Security Infrastructure, Globus' implementation of GSSAPI
GSSAPI	Grid Security Service Application Interface
HSM	Hierarchical Storage Management
Jiro	SUN Jiro, Implementation of the FMA specification
LDAP	Lightweight Directory Access Protocol
MDS	Monitoring and Discovery Service
MPI	Message Passing Interface
OCM-G	Grid-enable OMIS-Compliant Monitor
OMIS	On-line Monitoring Interface Specification
PVM	Parallel Virtual Messages
R-GMA	DataGrid relational Grid monitoring architecture
RPM	Red Hat Package Manager
SOAP	Simple Object Access Protocol
SRS	Software Requirement Specifications
VO	Virtual Organisation

3. REFERENCES

[1] Annex 1 – "Description of Work", CrossGrid annex1_v3.1.doc

[2] Application description including use cases for Task 1.1, CG-1-D1.1.1-UvA001.pdf
Application description including use cases for Task 1.2, CG-1.2.D1.2.1-003-SRS.pdf
Application description including use cases for Task 1.3, CG-1.D1.1.3-CSIC003.pdf
Detailed planning for air/sea pollution application including use cases, CG-1.4-DOC-ICM004-SRSusc.pdf

[3] General requirements and detailed planning for programming environment, CG-2-D2.1-0005-SRS.pdf, CG-2.2-DOC-USTUTT005-SRS.pdf, CG-2.3-DOC-UCY005-SRS.pdf, CG-2.4-DOC-CYFRONET001-SRS.pdf

[4] Portals and Roaming access, SRS_TASK_3.1.pdf

[5] Grid resource management, CG-3.2-SRS-0010.pdf

[6] Grid monitoring, Task3.3-SRS.pdf

[7] Optimisation of data access, CG-3.4-SRS-0012.pdf

[8] Test and Integration, CG-3.5-SRS-0002PUBLIC.pdf

[9] Detailed planning for Testbed Set up, CG-4-D4.1-001-PLAN.pdf

[10] State of The Art, StateOfArt.pdf, CG-3.4-STA-0010-StateOfTheArt.pdf, CG-3.5-SRS-0020-StateOfTheArt.pdf

[11] Definition of architecture, technical plan and evaluation criteria for scheduling, resource management, security and job description, DataGrid-01-D1.2-0112-0-3.

[12] Overview of common component and Grid services architectures, CG-5-TAT-ComponentsServices-001.doc

4. TEST SPECIFICATIONS

The tables shown in next subsections describe the test procedures proposed by each Task in order to check all components and tools delivered by WP3. This kind of tests will be applied from the beginning of Month 7 to Month 12.

4.1. TASK 3.1 PORTALS AND ROAMING ACCESS

The main mechanisms that will be taken into consideration are:

- Saving and restoring the Migration Desktop user profiles;
- Downloading/uploading files using Desktop GridFTP graphical client;
- Submitting job using Migrating Desktop;
- Submitting job using Application Portal;

<table>
<tr><td colspan="3">Test type: Saving and restoring of user profile.</td><td></td></tr>
<tr><td colspan="4">Application description</td></tr>
<tr><td colspan="4">a) General description of the application</td></tr>
<tr><td colspan="4">The subject of Task 3.1 is to establish a user friendly Grid environment by providing access to the Grid independently from the user location via portals or dedicated application – Roaming Desktop.</td></tr>
<tr><td colspan="4">b) The application architecture (modules, servers, clients, operation algorithm etc.)</td></tr>
<tr><td colspan="4">Task 3.1 provides following components:
• 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.
• 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, user authorisation and authentication, job submission, file transfer and grid monitoring
• LDAP Database – a network database used for storing user profiles;
• Migrating Desktop GUI – applet that provides “window” to the Grid environment.
The detailed description of these components and their mutual connections was provided in the first deliverable D3.1. This description can be found in the document CG-3.1-SRS-0017.doc [4].</td></tr>
<tr><td colspan="4">Requirements:</td></tr>
<tr><td colspan="4">a) Hardware:</td></tr>
<tr><td></td><td colspan="3">Computational power:</td></tr>
<tr><td></td><td></td><td>Number and architecture of the processors</td><td>1 or more x86 compatible processors</td></tr>
<tr><td></td><td></td><td>FLOPS</td><td>Not applicable</td></tr>
</table>

		Memory	Not applicable
	Storage (capacity required to store input, output, intermediate and measurement data for application)		
		Capacity	Not applicable
		Transfer rate	Not applicable
b) The network resources:			
		Type and number of interfaces	Not applicable
		Required bandwidth	Not applicable
		Latency (demanded/ acceptable)	Not applicable
		Firewall system configuration (required passage)	Each components will work with the SOAP protocol, and each component will have its own port (unprivileged ports), thus these components have to be available for each client, using authentication and authorization mechanisms.
c) Software:			
	Operating system (type, version) and patches		Linux Red Hat 6.2 or Windows family system for Migrating Desktop application Linux Red Hat 6.2 for Roaming Access Server – at least 1 server per domain
	Test application modules		Not applicable.
	Libraries (mathematical, graphical, security), licences, certificates		Security infrastructure used in the CrossGrid project probably will be GSI. In the case of components, which will use SOAP protocol it will be the GSI for SOAP. Web browser with Java plug-in v1.4 installed. Access to LDAP based database. CoG toolkit v 0.9.13
	Measurement software		Not applicable.
d) Required users' accounts on the systems			
1 user account with the elementary rights			
Description of the test procedure			
a) Test goals			
Checking the connections between all components and making elementary tests in the environment similar to the working one.			
b) General schema			
Loading Migrating Desktop with default settings Changing Migrating Desktop settings Closing Migrating Desktop Restarting Migrating Desktop.			
c) Measured parameters (efficiency, stability, correctness, bit rate, frames per second, latency, etc.)			
- Stability			

- Correctness		
d) Preparing to tests:		
	Accounts creating, software and patches installation and configuration (see 3.b and 3.c)	already described above
	Network interfaces and connections configuration, firewalls configuration etc. (see 3.a)	already described above
e) Tests:		
	Tests algorithm:	
	Executed programs (modules of tests application and measurement applications): locations, terms, order, dependencies	▪ LDAP database; ▪ Roaming access server; ▪ Migrating desktop application;
	Actions taken by the user	▪ Starting all necessary components. ▪ Loading Migrating Desktop with default settings ▪ Changing Migrating Desktop settings ▪ Closing Migrating Desktop ▪ Restarting Migrating Desktop ▪ Checking if all settings was correctly restored;
Results analysis		
a) Tests results analysis and interpretation		
	Determining the correctness of launching Migrating Desktop application with default settings.	Not measured yet
	Saving Migrating Desktop settings in LDAP based database,	Not measured yet
	Restoring user settings after restarting application.	Not measured yet
b) Other analysis		
Conclusions		
a) Saving and restoring of user profile mechanism.		
There are not conclusions.		
b) Other conclusions		
There are not conclusions.		

Test type: Transferring files using GridFTP graphical client.	
Application description	
a) General description of the application	
The subject of Task 3.1 is to establish a user friendly Grid environment by providing access to the Grid independently from the user location via portals or dedicated application – Roaming Desktop.	
b) The application architecture (modules, servers, clients, operation algorithm etc.)	
Task 3.1 provides following components: • 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. • 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, user authorisation and authentication, job submission, file transfer and grid monitoring • LDAP Database – a network database used for storing user profiles; • Migrating Desktop GUI – applet that provides "window" to the Grid environment. GridFTP graphical client is a part of Migrating Desktop GUI. The detailed description of these components and their mutual connections was provided in the first deliverable D3.1. This description can be found in the document CG-3.1-SRS-0017.doc [4].	
Requirements:	
a) Hardware:	
Computational power:	
Number and architecture of the processors	1 or more x86 compatible processors
FLOPS	Not applicable
Memory	Not applicable
Storage (capacity required to store input, output, intermediate and measurement data for application)	
Capacity	Not applicable
Transfer rate	Not applicable
b) The network resources:	
Type and number of interfaces	Not applicable
Required bandwidth	Not applicable
Latency (demanded/ acceptable)	Not applicable

		Firewall system configuration (required passage)	Each components will work with the SOAP protocol, and each component will have its own port (unprivileged ports), thus these components have to be available for each client, using authentication and authorization mechanisms.
c) Software:			
	Operating system (type, version) and patches		Linux Red Hat 6.2 or Windows family system
	Test application modules		Not applicable.
	Libraries (mathematical, graphical, security), licences, certificates		Security infrastructure used in the CrossGrid project probably will be GSI. In the case of components, which will use SOAP protocol it will be the GSI for SOAP. Web browser with Java plug-in v1.4 installed. Access to LDAP based database – OpenLDAP – 2.0.25 or above CoG toolkit v 0.9.13 Entry describing user account in Grid-mapfile on remote system. User valid proxy certificate. EDG VO support.
	Measurement software		Not applicable.
d) Required users' accounts on the systems			
1 user account with the elementary rights on workstation;			
Description of the test procedure			
a) Test goals			
Checking the connections between all components and making elementary tests in the environment similar to the working one.			
b) General schema			
Loading Migrating Desktop with default settings Launching GridFTP graphical client application; Connecting to remote GridFTP server; Putting local file on remote server; Getting file from remote server locally; Removing file from remote server;			
c) Measured parameters (efficiency, stability, correctness, bit rate, frames per second, latency, etc.)			
- Stability - Correctness			
d) Preparing to tests:			
	Accounts creating, software and patches installation and configuration (see 3.b and 3.c)		already described above

	Network interfaces and connections configuration, firewalls configuration etc. (see 3.a)	already described above
e) Tests:		
	Tests algorithm:	
	Executed programs (modules of tests application and measurement applications): locations, terms, order, dependencies	▪ Starting GridFTP server on remote location; ▪ Roaming access server; ▪ Migrating Desktop application; ▪ GridFTP graphical client
	Actions taken by the user	▪ Starting all necessary components. ▪ Loading Migrating Desktop with default settings ▪ Launching GridFTP graphical client ▪ Connecting to remote GridFTP server; ▪ Putting local file on remote server; ▪ Getting file from remote server locally; ▪ Comparing original file with uploaded and then downloaded one. ▪ Removing file from remote location.
Results analysis		
a) Tests results analysis and interpretation		
	Determining the correctness of establishing connection with remote server (authentication issues).	not measured yet
	Determining the correctness of uploading file.	not measured yet
	Determining the correctness of downloading file..	not measured yet
b) Other analysis		
Conclusions		
a) Transferring files;		
There are not conclusions.		
b) Other conclusions		
There are not conclusions.		

Test type: Submitting jobs using Migrating Desktop	
Application description	
a) General description of the application	
The subject of Task 3.1 is to establish a user friendly Grid environment by providing access to the Grid independently from the user location via portals or dedicated application – Roaming Desktop.	
b) The application architecture (modules, servers, clients, operation algorithm etc.)	
Task 3.1 provides following components: • 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. • 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, user authorisation and authentication, job submission, file transfer and grid monitoring • LDAP Database – a network database used for storing user profiles; • Migrating Desktop GUI – applet that provides "window" to the Grid environment. The detailed description of these components and their mutual connections was provided in the first deliverable D3.1. This description can be found in the document CG-3.1-SRS-0017.doc [4].	
Requirements:	
a) Hardware:	
Computational power:	
Number and architecture of the processors	1 or more x86 compatible processors
FLOPS	Not applicable
Memory	Not applicable
Storage (capacity required to store input, output, intermediate and measurement data for application)	
Capacity	Not applicable
Transfer rate	Not applicable
b) The network resources:	
Type and number of interfaces	Not applicable
Required bandwidth	Not applicable
Latency (demanded/ acceptable)	Not applicable

		Firewall system configuration (required passage)	Each components will work with the SOAP protocol, and each component will have its own port (unprivileged ports), thus these components have to be available for each client, using authentication and authorization mechanisms.
c) Software:			
	Operating system (type, version) and patches		Linux Red Hat 6.2 or Windows family system
	Test application modules		Not applicable.
	Libraries (mathematical, graphical, security), licences, certificates		Security infrastructure used in the CrossGrid project probably will be GSI. In the case of components, which will use SOAP protocol it will be the GSI for SOAP. Web browser with Java plug-in v1.4 installed. Access to LDAP based database – OpenLDAP – 2.0.25 or above CoG toolkit v 0.9.13 Entry describing user account in Grid-mapfile on remote system. User valid proxy certificate. Connection to CrossGrid WP1 application servers. CrossGrid Job Submission Service; DataGrid Resource Broker; EDG VO support.
	Measurement software		Not applicable.
d) Required users' accounts on the systems			
1 user account with the elementary rights on local host;			
Description of the test procedure			
c) Test goals			
Checking the connections between all components and making elementary tests in the environment similar to the working one.			
d) General schema			
Loading Migrating Desktop with default settings Defining job (choosing application, setting its parameters); Submitting job via Job Submission Service; Getting application output;]			
e) Measured parameters (efficiency, stability, correctness, bit rate, frames per second, latency, etc.)			
- Stability - Correctness			
f) Preparing to tests:			

	Accounts creating, software and patches installation and configuration (see 3.b and 3.c)	Already described above
	Network interfaces and connections configuration, firewalls configuration etc. (see 3.a)	Already described above
g) Tests:		
	Tests algorithm:	
	Executed programs (modules of tests application and measurement applications): locations, terms, order, dependencies	▪ Application servers on remote location; ▪ Roaming access server; ▪ Migrating desktop application;
	Actions taken by the user	▪ Starting all necessary components. ▪ Loading Migrating Desktop with default settings ▪ Defining job (choosing application, setting its parameters); ▪ Submitting job via Job Submission Service; ▪ Checking application state. ▪ Getting application output; ▪ Launching application once again. ▪ Canceling the application.
Results analysis		
a) Tests results analysis and interpretation		
	Determining the correctness of establishing connection with remote server (authentication issues).	not measured yet
	Determining the correctness of uploading file.	not measured yet
	Determining the correctness of downloading file..	not measured yet
b) Other analysis		
Conclusions		
a) Defining job;		
There are not conclusions.		
b) Submitting application;		
There are not conclusions.		
c) Receiving output.		
There are not conclusions.		
d) Other conclusions		

There are not conclusions.

Test type: Submitting jobs using Application Portals	
Application description	
a) General description of the application	
The subject of Task 3.1 is to establish a user friendly Grid environment by providing access to the Grid independently from the user location via portals or dedicated application – Roaming Desktop.	
b) The application architecture (modules, servers, clients, operation algorithm etc.)	
Task 3.1 provides following components: • 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. • 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, user authorisation and authentication, job submission, file transfer and grid monitoring • LDAP Database – a network database used for storing user profiles; • Migrating Desktop GUI – applet that provides "window" to the Grid environment. The detailed description of these components and their mutual connections was provided in the first deliverable D3.1. This description can be found in the document CG-3.1-SRS-0017.doc [4].	
Requirements:	
a) Hardware:	
Computational power:	
Number and architecture of the processors	1 or more x86 compatible processors
FLOPS	Not applicable
Memory	Not applicable
Storage (capacity required to store input, output, intermediate and measurement data for application)	
Capacity	Not applicable
Transfer rate	Not applicable
b) The network resources:	
Type and number of interfaces	Not applicable
Required bandwidth	Not applicable

		Latency (demanded/ acceptable)	Not applicable
		Firewall system configuration (required passage)	Each components will work with the SOAP protocol, and each component will have its own port (unprivileged ports), thus these components have to be available for each client, using authentication and authorization mechanisms.
c) Software:			
	Operating system (type, version) and patches		Linux Red Hat 6.2 or Windows family system
	Test application modules		Not applicable.
	Libraries (mathematical, graphical, security), licences, certificates		Security infrastructure used in the CrossGrid project probably will be GSI. In the case of components, which will use SOAP protocol it will be the GSI for SOAP. Web browser with Java plug-in v1.4 installed. Access to LDAP based database – OpenLDAP – 2.0.25 or above CoG toolkit v 0.9.13 Entry describing user account in Grid-mapfile on remote system. User valid proxy certificate. Connection to CrossGrid WP1 application servers. Connection to Application Portal Server. EDG VO support.
	Measurement software		Not applicable.
d) Required users' accounts on the systems			
1 user account with the elementary rights on local host;			
Description of the test procedure			
a) Test goals			
Checking the connections between all components and making elementary tests in the environment similar to the working one.			
b) General schema			
Connecting to WP1 Application Web portal Defining job (setting application parameters); Submitting job via Job Submission Service; Getting application output;			
c) Measured parameters (efficiency, stability, correctness, bit rate, frames per second, latency, etc.)			
- Stability - Correctness			
d) Preparing to tests:			

	Accounts creating, software and patches installation and configuration (see 3.b and 3.c)	Already described above
	Network interfaces and connections configuration, firewalls configuration etc. (see 3.a)	Already described above
e) Tests:		
	Tests algorithm:	
	Executed programs (modules of tests application and measurement applications): locations, terms, order, dependencies	▪ Application servers on remote location; ▪ Application Portal Server; ▪ Roaming access server; ▪ Web browser.
	Actions taken by the user	▪ Starting all necessary components. ▪ Connecting to WP1 Application portal; ▪ Defining job (choosing application, setting its parameters); ▪ Submitting job via Job Submission Service; ▪ Checking application state; ▪ Getting application output; ▪ Launching application once again; ▪ Canceling the application.
Results analysis		
a) Tests results analysis and interpretation		
	Determining the correctness of establishing connection with remote server (authentication issues).	not measured yet
	Determining the correctness of uploading file.	not measured yet
	Determining the correctness of downloading file..	not measured yet
b) Other analysis		
Conclusions		
a) Defining job;		
There are not conclusions.		
b) Submitting application;		
There are not conclusions.		
c) Receiving output.		
There are not conclusions.		

d) Other conclusions
There are not conclusions.

4.2. TASK 3.2 JOB SUBMISSION OVER THE CROSSGRID TESTBED USING A CG_SCHEDULING AGENT

Test type		**Job submission over the CrossGrid testbed using a CG_Scheduling Agent**
Application description		
a) General description of the application		
CG_Scheduling Agent (CGSA) is the middleware service in the CrossGrid project responsible for scheduling user jobs over the CrossGrid testbed infrastructure. For each job, it will look for available resources that could be used to run the job. Then, it will submit the job to a grid resource manager based on Condor-G, who will finally carry out all the necessary steps to run the job on the selected machines. CGSA will manage different kind of jobs that can be classified as sequential or parallel (MPI), according to the use of concurrency. From the point of view of their priority they will be classified mainly into two categories: batch jobs and interactive jobs. CGSA will run jobs on the available resources according to the priorities and preferences determined by the user for each job.		
b) The application architecture (modules, servers, clients, operation algorithm etc.)		
CGSA is an event-driven daemon that executes and infinite loop in which carries out the following operations: - it receives submission commands from a user through a web portal, - stores jobs into a queue that is sorted in order of priority, - contacts a CrossGrid Resource Broker (CGRB) to find suitable resources for each job in the queue, - submits each job to the corresponding resources provided by the Resource Broker, - contacts the grid resource manager to check the status of jobs running on grid resources and carries out the appropriate actions when a change in status is detected.		
c) Network traffic characteristic generated by the application:		
	Type of connections (point-point, 1 to many, many to many)	1 to 1 connection between two computers (user web portal and CGSA daemon). Although both elements may be running in the same machine. 1 to 1 connection between CGSA daemon and CGRB daemon. Both services could be also running in the same machine. 1 to many connections between the CGSA machine and the execution machines.

	Stochastic traffic characteristic (constant, variable steam etc.)	Variable steam.
	Short description of message passing algorithm	Not specified

Requirements:			
a) Hardware:			
	Computational power:		
		Number and architecture of the processors	Several x86 compatible machines
		FLOPS	Not specified
		Memory	At least 128MB each
	Storage (capacity required to store input, output, intermediate and measurement data for application)		
		Capacity	Not specified
		Transfer rate	Not specified
b) The network resources:			
		Type and number of connections	TCP/IP connections with a variable number of TCP ports used. 10/100 Ethernet boards.
		Required bandwidth	Not specified
		Latency (demanded/ acceptable)	Not specified
		Firewall system configuration (required passage)	Not specified
c) Software:			
	Operating system (type, version) and patches		Linux 6.2 (and/or 7.2)
	Test application modules		CGSA
	Libraries (mathematical, graphical, security), licences, certificates		Not specified
	Measurement software (used to estimate condition of network connections and computational servers load during tests)		test specific monitoring scripts
d) Required users' accounts on the systems			
User account on the local machine running CGSA. User needs also a valid entry in each gridmapfile of remote resources. Remote resources require also the existence of a pool of accounts for grid users. A scheduling test account with the same name, password and access rights in all resources would be desirable although this requirement has to be discussed with responsibles of the testbed WP as it may imply some potential security risks. The user needs also valid certificates and a grid proxy with an expiration time longer than 4hours. In addition to the existence of user accounts, one scenario of the test requires the existence of a set of local dedicated resources.			

e) Operator (people) accessibility in particular localisation (term, operations to do)
1 person on the local site to carry out the test. Correctly configured and working computers with basic CrossGrid middleware (which would include the basic services required from Globus and EDG).
f) Input data for test application
Set of simple executable applications, both sequential and parallel. Some of the applications will require also the existence of input data files.
Description of the test procedure
a) Test goals
Assess the correct execution of applications submitted to CGSA in the absence of errors in the grid environment. Assess that CGSA correctly holds jobs in the queue and provides an appropriate error warning in the presence of unexpected errors of any grid resource. Checking the average execution time and turnaround times for the applications in the test suit.
b) General schema
The test could be carried out in two modes: local and remote. In the local mode, each job will be submitted with a explicit specification of the local machines on which the application should run. This test should assess the correctness of application submission in the absence of errors. In the remote mode, each job will be submitted with a specification that requires the allocation of remote resources selected by the CGRB. For each one of the applications in the test suite the following steps will be followed: 1) The application will be submitted to CGSA and entered in his queue. 2) In the local mode, the job and the list of selected resources will be passed directly to Condor-G. On the remote mode, the ClassAd describing the job will be sent to the CGRB. If a suitable number of resources is provided by the CGRB, the job will be passed to Condor-G. Otherwise, the query to CGRB will be repeated a predefined number of times before the whole job is cancelled. 3) On application completion, CGSA will remove the job from the queue and a completion notification will be reported. For the sake of illustration, the test suite could be made of the following examples:

- a sequential program that simply spins in a loop for a programmable time period without any input/output operation (this is an example used as a batch job)..
- A sequential program that reads sequences of numbers from stdin and prints to stdout a modified sequence (this is an example used as an interactive job).
- A sequential program that reads lines from several input files and writes them into an output file.
- An MPI application without direct input/output.

In addition to the individual applications, the test suite will have a combined submission of several copies of pure batch jobs with a small number of interactive jobs to verify the ability of CGSA to preempt or suspend lower priority jobs and run the interactive ones in the released resources.

In the second year, the test suit may include also a synthetic application that models the behaviour of a representative application from WP1.

c) Measured parameters (efficiency, stability, correctness, bit rate, frames per second, latency, etc.)

- Summary time of computations,
- Number of machines used,
- Number of resubmissions in case of failures during the execution,
- time spent on the matchmaking process by the CGRB,
- overall delay introduced by CGRB,
- overall delay introduced by initial set-up of Condor-G.

d) Preparing to tests:

Accounts creating, software and patches installation and configuration (see 3.b and 3.c)	Already described above
Network interfaces and connections configuration, firewalls configuration etc. (see 3.a)	Already described above
Preparing input data (see 3.e)	Creating a copy of computations test application on the first system

e) Tests:

Tests algorithm:

Executed programmes (modules of tests application and measurement applications): locations, terms, order, dependencies	See above description of test suit.

		Actions taken by the user	User submits each application in the test suit form the Web Portal interface. Eventually, a simplified test script that does not use the web portal will be provided with the first prototypes of CGSA.
		Co-ordination with the other tests application	Not specified.
		Determining methods of collecting the working application parameters and the measurement results (collecting period)	Parameters are collected during the whole test in the CGSA log file and summarized at the end of the execution of each application on a summary log file.

4.3. TASK 3.3 GRID MONITORING

The products of Task 3.3 that will be tested are:

- an OMIS-based application monitoring system, OCM-G;
- additional services, SANTA-G, for ad-hoc non-invasive monitoring, and
- Jiro-based services for Grid-infrastructure monitoring.

Test type	**Application Monitoring using the OCM-G**
Application description	
d) General description of the application	
The Grid-enabled OMIS-compliant Monitor – the OCM-G – is an autonomous Grid monitoring service accessible via a standardized interface – OMIS (Online Monitoring Interface Specification). The OCM-G is aimed to be used by tools supporting application development from Task 2.4. Its purpose is to provide services for collecting information about (and possibly manipulating) applications.	
e) The application architecture (modules, servers, clients, operation algorithm etc.)	
The OCM-G is a distributed system composed of two types of components: 1. Service Managers which are daemons residing permanently, one per each site in the Grid; 2. Local Monitors which reside on each host of the Grid, one for each Grid user. Local Monitors are only started when they are needed, i.e., when there are application processes to be monitored on a given host.	

The functions of these modules are as follows. 1. Service Managers - route messages in the OCM-G - accept requests from tools - distribute the requests to Local Monitors - collect replies from the Local Monitors - assemble the replies and return a single reply to tools 2. Local Monitors - execute OMIS requests - accept requests from Service Managers and send replies back

f) Network traffic characteristic generated by the application:		
	Type of connections (point-point, 1 to many, many to many)	1 to 1 connection between a tool and a Service Manager (both may run on the same host) 1 to many connection between a Service Manager and Local Monitors many to many connection between Service Managers
	Stochastic traffic characteristic (constant, variable steam etc.)	Variable steam.
	Short description of message passing algorithm	Not specified

Requirements:			
a) Hardware:			
	Computational power:		
		Number and architecture of the processors	Approx. 10 PC machines
		FLOPS	Not specified
		Memory	Not specified
	Storage (capacity required to store input, output, intermediate and measurement data for application)		
		Capacity	Not specified
		Transfer rate	Not specified
b) The network resources:			
		Type and number of connections	TCP/IP connections with a variable number of TCP ports used.
		Required bandwidth	Not specified
		Latency (demanded/ acceptable)	Not specified
		Firewall system configuration (required passage)	Not specified

c) Software:		
	Operating system (type, version) and patches	Linux 6.2 (and/or 7.2)
	Test application modules	LM, SM
	Libraries (mathematical, graphical, security), licences, certificates	Not specified
	Measurement software (used to estimate condition of network connections and computational servers load during tests)	test specific monitoring scripts
d) Required users' accounts on the systems		
User account on a local machine. Pool of grid user accounts on which a LM executable is available. The user needs also valid Grid certificates.		
e) Operator (people) accessibility in particular localisation (term, operations to do)		
One person on the local site to carry out the test.		
f) Input data for test application		
A set of simple MPI parallel applications. Applications should be linked against a special (instrumented) version of the MPI library.		
Description of the test procedure		
a) Test goals		
Check the correctness of the OCM-G start-up procedure. Test the OMIS request distribution and assembly in the OCM-G. Test the correct working of services.		
b) General schema		
OCM-G start-up procedure In this test, a parallel MPI application composed of many processes spanning most available hosts should be submitted. The OCM-G part linked to the application should start the Local Monitors, which should contact their Service Managers. The application processes should register in LMs, which will pass the registration information to the SMs. On a local machine, the user runs a small (console) tool connected to the OCM-G via which he can send monitoring requests and receive the corresponding replies. The user sends request to get a list of running applications, and then the list of processes of the running application.		

Each application process may check its location and send it to one master process, which will record all the information in a file. In this or similar way, we can compare this information with the one received from the monitoring system.

OMIS request distribution and assembly

In this test, a similar application as in test 1. is needed. The user, on a local machine, uses a tool to submit monitoring requests and gathering replies. Each Local Monitor and each Service Manager should record all incoming requests (e.g. in a file). Thus, we may check if the distribution is correct. Similarly, the incoming replies should be recorded to check the correctness of request assembly.

Correct working of services

The test procedure in this case is strongly related to the particular service. For example, in case of request `thread_stop` which should stop the specified processes, we can record the state of involved processes in local files and then check if the correct processes were indeed stopped.

c) Measured parameters (efficiency, stability, correctness, bit rate, frames per second, latency, etc.)

- Correctness of the system start-up,
- Correctness of the algorithms of request distribution and assembly,
- Correctness of working of monitoring services,

d) Preparing to tests:

	Accounts creating, software and patches installation and configuration (see 3.b and 3.c)	To perform the test, an initial monitoring infrastructure has to be set up. The pool of hosts has to be divided into three groups (simulating different sites). For each group, one primary host has to be designated, on which a Service Manager has to be run. On each host, a configuration file must be available, in which the name of the primary host for the site is written.
	Network interfaces and connections configuration, firewalls configuration etc. (see 3.a)	Connection between monitoring components must be possible (firewall configuration). Ports TBD.
	Preparing input data (see 3.e)	Not specified

e) Tests:

	Tests algorithm:

		Executed programmes (modules of tests application and measurement applications): locations, terms, order, dependencies	See above description of test suit.
		Actions taken by the user	User submits the parallel applications from the local account and uses the tool to submit monitoring requests to the OCM-G and gather replies. The requests may be prepared in a script to ease the test procedure.
		Co-ordination with the other tests application	Not specified.
		Determining methods of collecting the working application parameters and the measurement results (collecting period)	The log files created on hosts should be collected and delivered to the local machine.

Test type	**Use of SANTA-G with the CrossGrid testbed**
Application description	
a) General description of the application	
SANTA-G is an extension to the DataGrid R-GMA grid information system. It enables the introduction of non-invasive monitoring information from external instruments into the R-GMA, specifically to support ad-hoc rather than routine monitoring. For each monitoring experiment the user will be responsible for the establishment of the external instrument environment. It is assumed that the instruments will have their own control software, and that they will generate a set of one or more result files to which fast local access is possible. It is also assumed that in the worst case the result files will be very large. SANTA-G is specifically designed to enable relational query of these files in-place, i.e. without data migration or more than read-only sharing. For each experiment, initializing SANTA-G will establish a connection to both a standard R-GMA DBProducer servlet and one or more specialized CanonicalProducer servlets, each of which will yield relevant entries in the R-GMA registry. Thereafter SANTA-G will accept queries from the R-GMA, and will satisfy these using a query engine executing in the local environment of the result files. The result sets will be returned to the R-GMA. For the CrossGrid testbed, SANTA-G will be demonstrated with a net tracer and an Ethernet Packet Viewer. The instrument software will be standard TCPDump, which generates network trace files.	

b) The application architecture (modules, servers, clients, operation algorithm etc.)

On the instrument [producer] side, SANTA-G consists of a Sensor, a Canonical Producer, a Canonical Producer Servlet and a Canonical Query Engine, which carry out the following operations:

- the user starts the monitoring experiment by starting the Sensor;
- the Sensor reads the user's configuration file, initialises a DBProducer [which connects to a DBProducer servlet and registers with the R-GMA], and stores configuration values in it;
- the Sensor then initializes one or more Canonical Producers [each of which connects with a Canonical Producer Servlet and registers itself with the R-GMA];
- the Sensor starts the net tracer's TCPDump, thereby [possibly continuously] creating result files;
- external queries arrive at the DBProducer as a result of lookups of the R-GMA registry, and are executed by it's database engine, which returns resultsets to the R-GMA;
- external queries arrive at the Canonical Producer as a result of querying the DBProducer, and are executed by it's Canonical Query Engine, which returns resultsets to the R-GMA.

On the viewer [consumer] side, SANTA-G consists of a Viewer GUI and a Consumer, which carry out the following operations:

- the user starts the Viewer, which initialises the Consumer [which connects to a Consumer servlet];
- the user enters relational queries into the GUI, either directly or via navigation using the GUI;
- the Consumer forwards the query [via the Consumer servlet] to the R-GMA, which finds [by using its Mediator to look up the registry] the SANTA-G DBProducer, which then executes the query and returns a resultset containing the URI of the Canonical Producer;
- The Consumer forwards the query to the R-GMA, for direct execution by the Canonical Producer [using the Canonical Producer Engine], which returns resultsets to the R-GMA;
- The GUI displays the resultsets.

c) Network traffic characteristic generated by the application:

	Type of connections (point-point, 1 to many, many to many)	1 to 1 connection between Producer /Consumer and Producer/Consumer servlet, although both elements may be running in the same machine. 1 to many connections between a Canonical Producer servlet and many Canonical Producers, although both elements may be running in the same machine. 1 promiscuous network connection from a network switch to the net tracer.
	Stochastic traffic characteristic (constant, variable steam etc.)	Variable steam.
	Short description of message passing algorithm	Not specified.

Requirements:

a) Hardware:

	Computational power:		
		Number and architecture of the processors	Several x86 compatible machines .
		FLOPS	Net tracer: at least a 2GHz Pentium4. Other machines: not specified.
		Memory	Not specified.
	Storage (capacity required to store input, output, intermediate and measurement data for application)		
		Capacity	Net tracer: at least 500GB for trace files. Other machines: not specified.
		Transfer rate	Not specified.
b) The network resources:			
		Type and number of connections	TCP/IP connections with a variable number of TCP ports used. 10/100/1000Mbps Ethernet boards (2 on net tracer). Network switch supporting promiscuous connection.
		Required bandwidth	Not specified.
		Latency (demanded/ acceptable)	Not specified.
		Firewall system configuration (required passage)	Not specified.
c) Software:			
	Operating system (type, version) and patches		Linux 6.2 (and/or 7.2).
	Test application modules		SANTA-G.
	Libraries (mathematical, graphical, security), licences, certificates		Not specified.
	Measurement software (used to estimate condition of network connections and computational servers load during tests)		Test-specific monitoring scripts.
d) Required users' accounts on the systems			
User account on the local machine running SANTA-G. User needs also a valid entry in each grid-mapfile of remote resources, or remote resources must support a pool of accounts for grid users. The user needs also a valid certificate and grid proxy.			
e) Operator (people) accessibility in particular localisation (term, operations to do)			
1 person on the local site to carry out the test. Correctly configured and working computers with base EDG middleware, TB1.2 or later.			
f) Input data for test application			

TCPDump will generate network trace files.

Description of the test procedure

a) Test goals

Assess the correct execution of SANTA-G components in the absence of errors in the grid environment.

Assess that SANTA-G correctly responds and provides an appropriate error warning in the presence of unexpected errors at any grid resource.

Checking the average execution times for relational queries in the test suite.

b) General schema

The tests will be carried out in three acquisition modes [*off-line*, *snapshot* and *continuous*] and two configuration modes [*open* and *closed*]. All combinations will be evaluated.

Off-line means a set of network trace files will be acquired in advance of the tests; *snapshot* means that one set will be acquired at the start of the tests, whilst *continuous* means they will be acquired continuously as the testing takes place.

Open means that SANTA-G will run on the net tracer [i.e. the net tracer will be an *open* external instrument], while *closed* means it will run on a machine that has read-only shared access to the net tracer's trace files.

For each test the following steps will be followed:

4) The SANTA-G Sensor and Viewer will be started, in a range of temporal orderings.
5) A test suite of relational queries will be issued from the Viewer.
6) The resultsets will be evaluated.
7) Timing and error-related data will be evaluated.

A range of queries, from simple to very complex, returning small to large resultsets, will be executed.

To assist testing, and as a useful facility for users, the Viewer will be enabled to log received resultsets in a user-defined logfile if so desired.

For off-line and snapshot acquisition the logged resultsets will be evaluated for correctness in relation to the trace files [which will still exist] after the test has completed. For continuous acquisition the slot mechanism will be modified to ensure retention of all generated trace files, so that the resultsets can be evaluated for correctness in relation to the trace files after the test has completed.

Early testing will be performed with 10Mbps and then 100Mbps networking, with only one active Viewer, in a synthetically unloaded CrossGrid testbed environment. Later this will be extended to 1Gbps networking,

and/or multiple active Viewers, and a live testbed environment.		
c) Measured parameters (efficiency, stability, correctness, bit rate, frames per second, latency, etc.)		
- Correctness, - completeness, - error-related data, - time spent on executing the queries at the Canonical Query Engine, - time from query issue to resultset arrival.		
d) Preparing to tests:		
	Accounts creating, software and patches installation and configuration (see 3.b and 3.c)	Already described above.
	Network interfaces and connections configuration, firewalls configuration etc. (see 3.a)	Already described above.
	Preparing input data (see 3.e)	Already described above.
e) Tests:		
	Tests algorithm:	
	Executed programmes (modules of tests application and measurement applications): locations, terms, order, dependencies	Already described above.
	Actions taken by the user	Already described above.
	Co-ordination with the other tests application	Not specified.
	Determining methods of collecting the working application parameters and the measurement results (collecting period)	Already described above.

(TBD Jiro-based test procedures)

4.4. TASK 3.4 DATA STORAGE AND RETRIEVAL

Test type: Data storage and retrieval (visualisation, distributed computations, transfer of big amounts of data, etc.)		
Application description		
a) General description of the application		
Task 3.4 – 'Optimization of Data-Access' provides services to support users in their interaction with large data manipulation; we propose a kind of expert system for sophisticated local data-handlers selection which allows building flexible environments for data storage and retrieval. Additionally, within this task a common middleware for fast tape file access and a system for data access time estimation will be developed.		
b) The application architecture (modules, servers, clients, operation algorithm etc.)		
Task 3.4 provides following components: - Data Access Estimator (DAES) – it estimates the data-access cost (i.e. latency and bandwidth) inside storage elements. - Component Expert Subsystem (CEXS) – it is the software-engine working according to Component-Expert Architecture; selecting components depending on the call-context. (These components are slightly different from components are listed here. They are the special kind of components, which in the case of the CrossGrid project will be worked as data-handlers) - Tape Resident Large Files Middleware (TRLFM) – it is the middleware placed on the top of existing HSM systems for speedup read access to large files (>100MB) stored on tapes, by means of reducing the latency time. - GridFTP Plugin (GPLG) - it is the plug-in to the GridFTP Server taking control over GridFTP server and passing on some requests to other components especially to CEXS. - Replica Manager (REMG)- it is prepared by the DataGrid project and only adopted by Task 3.4. It is the component, which is especially responsible for the replica selection and management - Storage Element (STEL) - Storage Element is responsible for publishing current configuration of a storage and answering on general questions about current configuration. All above components are working in the client-server architecture. All of them are the servers and some are also the clients (e.g. DEAS is the client of CEXS) The detailed description of these components and their mutual connections was provided in the first deliverable D3.1. This description can be found in the document CG-3.4-SRS-0012 [7].		
c) Network traffic characteristic generated by the application:		
	Type of connections (point-point, 1 to many, many to many)	The components DAES, GridFTP server, REMG, STEL use 1 to many model. The components GPLG and TRLFM use the point to point model.
	Stochastic traffic characteristic (constant, variable steam etc.)	Both models are appropriate
	Short description of message passing algorithm	Not applicable
Requirements:		

a) Hardware:			
	Computational power:		
		Number and architecture of the processors	1 or more x86 compatible processors
		FLOPS	Not applicable
		Memory	128 MB min
	Storage (capacity required to store input, output, intermediate and measurement data for application)		
		Capacity	100 GB
		Transfer rate	5 MB/s
b) The network resources:			
		Type and number of interfaces	At least 100 Mb/s Ethernet interface for each machine
		Required bandwidth	Not applicable
		Latency (demanded/ acceptable)	Not applicable
		Firewall system configuration (required passage)	Each components will work with the SOAP protocol, and each component will have its own port (probably port number over 2048), thus these components have to be available for each client, using authentication and authorization mechanisms.
c) Software:			
	Operating system (type, version) and patches		Linux Red Hat 7.1
	Test application modules		Not applicable.
	Libraries (mathematical, graphical, security), licences, certificates		Security infrastructure used in the CrossGrid project probably will be GSI. In the case of components, which will use SOAP protocol it will be the GSI for SOAP.
	Measurement software (used to estimate condition of network connections and computational servers load during tests)		Not applicable.
d) Required users' accounts on the systems			
5 users' accounts with the elementary rights			
e) Operator (people) accessibility in particular localisation (term, operations to do)			
1 person on PSNC side to perform test.			
f) Input data for test application			
Description of the test procedure			
a) Test goals			
Checking the connections between all components and making elementary tests in the environment similar to the working one. Testing the component selection made by CEXS.			
b) General schema			
Inserting a new data into a storage element			

Make manually replica of files Access to the data using GridFTP protocol. (Each storage element must run its own GridFTP server) Checking the logs prepared by the components, especially by CEXS.		
c) Measured parameters (efficiency, stability, correctness, bit rate, frames per second, latency, etc.)		
- Stability - bit rate - latency - general efficiency		
d) Preparing to tests:		
	Accounts creating, software and patches installation and configuration (see 3.b and 3.c)	already described above
	Network interfaces and connections configuration, firewalls configuration etc. (see 3.a)	already described above
	Preparing input data (see 3.e)	Creation of a copy of the sample data provided by the task 3.4. Installation of sample components for the Component-Expert Architecture and rules for the expert system nested in CEXS.
e) Tests:		
	Tests algorithm:	
	Executed programs (modules of tests application and measurement applications): locations, terms, order, dependencies	Each component will have its own starting script.
	Actions taken by the user	Starting all necessary components. Using GridFTP server.
	Co-ordination with the other tests application	Components provided by this task are connected with the portal provided by the task 3.1. Thus, the tests for tasks 3.4 and 3.1 should be coordinated.
	Determining methods of collecting the working application parameters and the measurement results (collecting period)	The components will provided auto-logging system, which will allow us to make analysis of the internal work each of them. The bit rate should be measured with 1 second interval.
Results analysis		
a) Tests (measurements) results analysis and interpretation		
	Determining the maximum, minimum, average and medium values of parameters defined in 4.b (quality indexes and the other statistical parameters of the application work) during stable network work	not measured yet
	Taking into consideration machines load and state of the network connections during the tests	not measured yet

	Evaluation of the influence of the events in the test network on the application quality parameters (what their changes are and their stochastic parameters) e.g.		
		Influence of incidental and periodical events	not measured yet
		Influence of events in different situations (e.g. with different network load)	not measured yet
		Influence of the other events	not measured yet
b) Analysis of aggregated results (for all test procedures or for simultaneously executed tests)			
	Evaluation of mutual influences of the simultaneously executed tests		not measured yet
	General utility features following all tests qualification		not measured yet
c) Other analysis			
Conclusions			
a) Conclusions of the network utility features:			
There are not conclusions.			
b) Conclusions concerning the router configuration that is optimal for the application in term of its specific usage parameters			
There are not conclusions.			
c) Other conclusions			
There are not conclusions.			

5. DEPENDENCY DESCRIPTION

This section provides information concerning the dependencies and interactions of this Task with other CrossGrid Work Packages and WP3's Tasks.

Dependencies and interactions with WP4 (Testbed Set up)

The official prototype version of the whole set of tools of the WP3 – middleware is prepared with the aim of running on an interactive Grid framework. For this reason, Task 3.5 is very strongly correlated with the prototypes released in WP4.

Testbeds are a key element in Grid Projects. They provide the framework to test and run the applications and corresponding middleware in a realistic distributed environment. The CrossGrid testbed will largely benefit from the DataGrid experience on testbed set up. Initial testbeds required by Task 3.5 follow the EDG scheme, running on the IA32 platform with Linux (RH6.2), assuring interoperability and an adequate framework for early checks of the infrastructure and organizational issues [9]. In a second phase, testbeds will improve the support for interactive distributed applications, and the corresponding middleware developed inside the project and WP3.

The first testbed set up will take place at sites with experience in the EDG software installation and where hardware is already installed and available. The list of sites includes IFCA (Santander) and IFIC (Valencia) for CSIC (Spain), FZK (Karlsruhe, Germany), LIP (Lisbon, Portugal), and TCD (Dublin, Ireland). All these sites have a member in the Integration Team for the WP4. This Integration Team is the basic technical force to manage the testbed releases. One of the objectives is the collaboration with the EDG Integration Team. The full Integration Team includes also technical representation from WP1 responsible for application deployment (contact is Dick van Albada), from WP2 (contact is Roland Weissmuller), and WP3 (Santiago González). The initial set up for this testbed sites should be ready by July 2002. All sites should have the initial testbed installed before September 2002.

The WP3 Integration Team [8] is collaborating with the WP4 Integration Team. A first meeting took place at CERN (July 2002), to get an in-site knowledge of the EDG testbed 1 set up, and to review issues regarding testbed set up and computing centre operation. One objective for the WP3 Integration Team is an evaluation of Globus packaging scheme and DataGrid software. This is clearly a vital evaluation with serious consequences for the success of the testbed and of the WP3 installation software.

To produce the first release of the WP3 software, the members of the WP3 Integration Team should be physically in the same place and have enough resources to build a small "grid". FZK Karlsruhe will provide resources needed for the intensive integration phase [8]. We will use the Karlsruhe testbed set up and one or two people of the WP4 Integration Team will take part in integration phase, to be a "bridge" between WP3 and WP4.

The WP3 and CrossGrid software will be installed from the CVS server. This central CrossGrid repository is in Karlsruhe with read-write access to repositories of WP1, WP2 and WP4. The read-write repository of WP3 is localized in Poznan with a read access to CrossGrid repository in Karlsruhe and will be updated on a daily basis. Task 3.5 will provide a backup copy for repositories located in Valencia.

Dependencies and interactions with WP1 (CrossGrid application development)

The application Tasks (WP1) will not need components from WP2 and WP3 in the first project year, but will be prepared to integrate these components for testing after one year. For the first year, the only common development identified is the datamining that is shared between tasks 1.3 and 1.4. The tools developed in Tasks 3.1 and 3.4 will be used directly by the applications under the scope of optimised data access and simplified user access to each of the applications deployed in WP1 [2]. Later on, most of the applications will need the visualisation toolkit. The roaming access tool will directly support the end user by making movement easier in the Grid environment.

Dependencies and interactions with WP2 (Grid application programming environment)

WP2 will analyse the types of run time data needed by the tools to be developed in WP3 and examine which of the needed data is already available via the DataGrid services [3]. Since the access to all run time data will occur through the unified monitoring interface developed in Task 3.3, the results of the analysis will define the major requirements for this [6].

The monitoring system to be designed within Task 3.3 will provide information from the three major sources performance data: applications, instruments, and infrastructure. Application information will be obtained by OCM-G. Specialised application monitors embedded in the application address space will provide dynamic application data, such as lists of running processes, CPU loads etc. It will also allow for the manipulation of applications, starting and stopping processes, etc. The primary user of this data will be Task 2.4 within WP2. Task 2.4 is developing "Interactive and semiautomatic performance evaluation tools". They require access to dynamic application data in order to accurately predict future performance and to provide performance measurements. This data will also be used by Task 3.2, responsible for "Grid resource management", in order to predict near future requirements and to guide scheduling actions.

Dependencies and interactions with WP3's Tasks

The deliverables from each of the software Tasks is expected in Month 10. Each partner should test well its own software before releasing the code to the WP3 Integration Team. It is very important to guarantee that all interfaces will be defined correctly. In July each partner sent a document to the Integration Team describing the test procedures and specifications (algorithms, devices, use cases, etc..) for each module (see section 4). The testbed supported by WP4 should be in place (in each site) with Globus and DataGrid's tools installed, where each Task test its tools and services. Each partner will provide test results and detailed specifications for any public package they provide or require (API, protocol, services, etc.) to the WP3 Integration Team.

A CVS server has been installed in Poznan (for access to the CVS server WP3 members should send a e-mail to miron@man.poznan.pl for a password and further details). Tasks in WP3 are invited to move their source repositories to this server at their convenience.

Task 3.1 Portals and roaming access

Task 3.1 will support the end user and is strictly correlated with WP1, giving an added value feature at the application level. Task 3.1 should use an efficient data access mechanism. Therefore they will also have connection with Task 3.4 using Replica Manager [7], and Task 3.2 using Resource Broker [5]. There is not direct connection between Task 3.1 and 3.3. Task 3.1 should use an efficient data access mechanism, therefore it will also use the results and mechanism developed in Task 3.4. The design and development phases of the roaming access service and portal approach will be done mainly by partners (PSNC, DATAMAT, ALGOSYSTEM, UCY). The integration and test phase will be mainly realized by Task 3.5 for M 10 –12, using the testbed delivered by WP4 in Karlsruhe.

Task 3.2 Grid resource management

Task 3.2 is developing a Grid resource management system based on self-adaptive Scheduling Agents for scheduling a particular parallel application submitted to a Grid. The Scheduling Agents are schedulers that make decisions about where, when, and how to run parallel jobs as specified by a set of policies, priorities, requirements and limits [5]. They are responsible for deciding the allocation of application Tasks to computing resources in a way that tries to guarantee that applications are conveniently, efficiently and effectively executed. Agents will interact with other services developed in the CrossGrid project, and their specification is aimed to be compatible with other on-going grid projects (e.g., DataGrid).

Scheduling Agents extend the basic services of the Workload Management System (WMS) to user jobs consisting of parallel applications that may, eventually, exhibit an interactive behaviour. Initially,

support will be provided to parallel applications using MPI, as this is the original platform described in the project proposal which is also mentioned in most of the SRS documents (from the deliverables of month three) prepared by the corresponding tasks. Support for other parallel libraries (such us, PVM) or environments (Cactus) may be discussed in the future according to the special needs that WP1 partners will report. For Prototype 0 (Month 12) and integration phase, Scheduling Agents will be designed to work with simple jobs. Additional support for composed jobs will be added at later stages of the project, once the proposed syntax to represent this kind of jobs has been discussed and approved with WP1 developers. The architecture and interface design of scheduling agents is being developed mainly by UAB and DATAMAT and CSIC will collaborate on the design and implementation of particular agents for certain parallel programming paradigms.

Task 3.3 Grid monitoring

The main system interfaces for Task 3.3 are to the Grid resource management component, Task 3.2, optimisation of DataAccess component, Task 3.4, and performance evaluation tools, Task 2.4 [6]. The application-level monitoring environment will comprise an autonomous monitoring system. For communication with tools, the monitoring system will comply to a standardised interface. The efficiency of application monitoring will be provided by specialised application monitors, embedded in the application address space, and by efficient performance data storage. The issue of scalability will be addressed by distributing the monitoring system into service managers (intermediate layer between tools and local monitors), local monitors, and application monitors. All partners of the Task 3.3 are working on the design specification of the monitoring tools (CYFRONET, ICM, TCD) and more general tests and integration will be realized within Task 3.5.

Task 3.4 Optimisation of data access

A kind of expert system, Data Access Package (DAP), is being developed by Task 3.4 in order to help users in their interaction with data store and to obtain an efficient access to the data [7]. This expert system for data management will be used by the Grid resource management system (Task 3.2). All systems from this Task are being designed in order to co-operate with the portal described in Task 3.1. Since a similar area is covered by WP2 of the DataGrid, this Task is trying as far as possible to avoid unnecessary duplication of major middleware features by staying aware of their work and collaborating as closely as possible. The development of this service tool is mainly done to CYFRONET. The test and integration phase will be realised for Month 10 – 12 within Task 3.5 by other partners (PSNC, UAB and CSIC).

6. STATUS OF THE INTEGRATION PROCESS (IN MONTH 6)

The integration and general testing will begin in Month 10 and will continue for approximately 2 months [8]. The deliverables from each of the software Tasks are expected in Month 10. Each partner should test well its own software using a local testbed before releasing the code to the Integration Team. This software should be obtained from WP3 repository localized in Poznan and updated in Karlsruhe (central CrossGrid repository) on day cycles.

To produce the first functioning release of the WP3 software, all of the members of the WP3 Integration Team will be physically in the same place and have enough resources to build a testbed supported by WP4 [9]. This testbed will be used as a testing environment, allowing testing separate tools (Tasks: 3.1 … 3.4) as well as the whole prototype of WP3 middleware. We will use part of the testbed developed in FZK Karlsruhe to do these testing and integration process. One or two people from the WP4 Integration Team will take part in integration phase.

In this testbed the reference platform (Red Hat Linux 6.2) has to be installed and verified on all machines. Globus and DataGrid software installations should be tested in this set of machines (the configuration of this mini-farm has to represent typical examples from testbed sites).

The Integration Team should install WP3 software from the CVS server and check the compatibility of this set of tools and services with Globus and DataGrid software. Then apply the same test procedures followed by each partner before realising its own software and test the dependencies between WP3 Tasks (see section 4). Finally, the Integration Team should produce the first beta-release of the WP3 software (using RPM files) and prepare enough documentation about the installation process. The Integration Team should consider a formal tutorial or training for the remote site administrators. Task 3.5 and the Integration Team will also provide a central HTTP (for example WP3 Web page in Poznan) repository for the packaged source and binary distributions of the whole WP3 software. The source distributions will contain the exact version of the code used to build the corresponding binary distribution.

During the first three months each Task defined the technical infrastructure (hardware and software) necessary for the development of these tools in close collaboration with the CrossGrid Architecture Team [4, 5, 6, 7, 8]. They also specified interfaces between the different components developed, defined the monitoring environment functionality and interfaces to Task 2.4 and defined the specification of GRM agents. Each Task delivered the information to WP4 for initial infrastructure set up [9] and has defined requirements from the applications in WP1 [2] and WP2 [3]. Also a review of existing technologies for the tools to be developed in this workpackages was done [10].

Now (Month 4 – 6) each partner is working on designing the architecture of the new grid services and tools, interfaces between tools and specifying security issue (e.g.,: design the secure communication between tools, specifying ways of secure authentication and authorisation, proposing standards of implementing security features). Each partner is preparing a document about test procedures to validate its own software (see section 4), which will be sent to the WP3 Integration Team. This document contains details concerning the test phase for each Task [3.1 … 3.4] (see section 4). Task 3.5 has to concentrate in this period on module tests made by each Task and tests of dependencies between WP3 Tasks in order to see if the integration process is possible. A list of interfaces for interoperability between modules within our Task, technology and tools is included in the D3.2 deliverable for month 6.

The CrossGrid project addresses technical challenges like interactive response, real distributed processing in parallel across different sites, access and data-mining on distributed databases, etc. Most of these requirements will be satisfied with the use of adequate middleware developed in our WP. The

Architecture Team is defining these requirements on terms of services implemented, having in mind the next Globus release, Globus3, that will be based on the OGSA (Open Grid Services Architecture). The CrossGrid testbed will have consequently to support these services and elements. Also, the CrossGrid testbed will largely benefit from the DataGrid experience on testbed set up. The WP3 is developing new Grid services and tools, which have not been addressed in other Grid projects and will be validated and tested thoroughly on the CrossGrid testbeds. The work has to be done in close collaboration with the Grid Forum and the DataGrid project to profit from their results and experience, and to obtain full interoperability. This will result in further extension of the Grid across Europe.

For these reasons, we need to assure compatibility between the whole set of tools of the WP3 and other Grid projects, like DataGrid or EuroGrid, and Globus middleware.

Interoperability and compatibility of Task3.1's components with DataGrid and Globus projects

Some mechanisms developed in DataGrid will be used by Task 3.1:

- Replica Manager (DataGrid WP2), (CrossGrid Task 3.4);
- Replica Catalogue (DataGrid WP2);
- File Copier (DataGrid WP2);
- Storage Element (DataGrid WP5);
- Resource Broker (DataGrid WP1), (CrossGrid Task 3.2);
- Job Submission Service (DataGrid WP1);

The client (Web Browser) orders downloading a file from some Grid localisation to a local workstation to HTTP Portal Server, and this server forwards it to the Roaming Access Server. The Roaming Access Server asks the Replica Manager about the location of the best replica and the Roaming Access Server, - based on the answer from the Replica Manager, orders the Replica Manager to copy the file operation from some Storage Element to the local host – the Roaming Access acts only as the mediator in the transmission of files.

All information needed to run a job will be passed to the Resource Broker using the Job Description Language (e.g. job name, input arguments, job requirements as well as input and output data files localisation). The user can run an application with the input data files that are normally stored on the local workstation or on any of the Grid Storage Elements. The Resource Broker will transfer files to the remote Computing Element.

The future popularity of the CrossGrid applications and tools depends on the functionality of the front-end. The proposed protocols mentioned by Task 3.1 for that part of the product are SOAP (Single Object Access Protocol), Globus GSI (Grid Security Infrastructure) and LDAP (Lightweight Directory Access Protocol).

Interoperability and compatibility of Task 3.2's components with DataGrid and Globus projects

Scheduling Agents developed in Task 3.2 constitute an extension of part of a Workload Management System (WMS) as described in the DataGrid project [11] . Submitted jobs will be described by job-ads using the Job Description Language (JDL) or ClassAd terminology. Scheduling Agents will require also access to information about the available resources in the Grid environment that may be used to service each user request. Resource information will also be expressed in terms of ClassAds (as used in EDG). The necessary extensions will be added to the EDG's JDL to be able to represent the requirements of parallel applications and computing resources in order to carry out the selection process. Thus, compatibility with EDG will be achieved by the use of a common JDL based on Condor ClassAds. Additionally CrossGrid Resource Broker will have an interface to MDS servers that use EDG schema. Potentially, later releases of CrossGrid Resource Broker will be interfaced to other information services developed at EDG.

Scheduling Agents have been designed to have a second interface that will be used to pass the list of selected resources to the grid resource manager that is going to be responsible of the actual execution of the job on the selected resources. In particular, a first interface will be implemented to be compatible with the DataGrids's Job Submission Service (JSS) to receive asynchronous notifications from the JSS about relevant events concerning a job, so that the Scheduling Agents can take appropriate actions.

Internal interfaces between the elements included in a Scheduling Agent are been described in the deliverable D3.2, together with their internal architecture. According to future developments of Globus, communication interfaces of Scheduling Agents will be compatible with Globus services based on the OGSA specification.

The Submission Service will be a simple wrapper that will finally submit the job to a grid resource manager. Condor-G is going to be adopted as grid resource manager because it provides the desirable characteristics of reliability and persistency required to support temporal failures in different elements of the grid infrastructure (Computing Elements, Globus gatekeepers, etc..).

Scheduling Agents will rely, in general, on EDG middleware, which also relies on Globus and Condor services. They will be based on Condor-G as a basic grid resource manager. Condor-G will ensure compatibility of Scheduling Agents with all Globus services related to job execution. Moreover the CrossGrid Resource Broker will have an interface to MDS servers that use the Globus schema. Task 3.2 will reuse some elements developed in the WMS of EDG. The use of common services should guarantee also the compatibility of CrossGrid Scheduling Agents with DataGrid jobs (i. e. any DataGrid jobs submitted to a CrossGrid Scheduling Agent should run on a CrossGrid testbed just as any CrossGrid job).

Integration and compatibility with other Tasks will be achieved through the use of a common JDL to describe CrossGrid jobs. The main interaction of Scheduling Agents with other services is through the user web portal (developed in Task 3.1), which is the mechanism used to submit jobs to Scheduling Agent. Integration to other services will be achieved through the user web portal.

Interoperability and compatibility of Task 3.3's components with DataGrid and Globus projects

CrossGrid will use the Globus software, and therefore the initial testbed will use the Globus MDS information system, which is due to become obsolete at the end of 2002. CrossGrid will also use the DataGrid software, and hence the DataGrid R-GMA information system. DataGrid have expended a lot of effort in formulating their MDS schema and populating it for their Test Bed 1, and creating a compatible R-GMA that will become the primary information system for their Test Bed 2.

Task 3.3 instead proposes three ways to create more information about the current state of the Grid. The first will monitor Grid applications at runtime in a OMIS compliant manner. The second will non-invasively monitor Grid components, and create a relational trace database of this information. A final activity will monitor infrastructure components using Jiro technology. These flows will supplement the existing information system content rather than involve design of a new information system. For the MDS, an appropriate extension to the DataGrid MDS schema will need to be formulated and populated with details of the CrossGrid testbed (prototype in Month 12). This should also be done for the R-GMA.

The consequence of this is that, at least for CrossGrid testbed (prototype in Month 12), a significant proportion of the information required by the other Tasks and workpackages can then be satisfied by the existing information systems. A further consequence is that these information systems become available for use by the new information sources. Where useful OMIS, result and trace information will be routed through them, and result and trace information can use the persistent database provision of the R-GMA. The Jiro information might be reflected into them, in much the same way as MDS currently is reflected into R-GMA. Task 3.3, in general, will make use of Globus.

The main assumption in Task 3.3 is the use of DataGrid R-GMA information system by the Grid Monitoring components. The position should be re-evaluated over time as new Grid information systems become available, e.g. OGSA compliant systems.

Interoperability and compatibility of Task 3.4's components with DataGrid and Globus projects

The Data Access Package (DAP) developed in Task 3.4 is an extension of the available tools developed within Globus and DataGrid projects mainly, but other ideas have been and will be incorporated as well. Task 3.4 is going to extend the functionality of the EDG Replica Manager in domain of the best replica selection of the data stored in databases. The component-expert subsystem, data time access estimator, optimisation of access to large tape resident files and the extension of EDG Replica Manager are the added value of the Task 3.4. The first three items represent low-level elements, not exactly grid services, however they are new tools that can be useful in the grid environment. The last item is the extension of the EDG activity in data management.

The DAP is being developed with the assumption that Globus version 3.0 and OGSA architecture will be future standards in such solutions. Thus, all architecture efforts are verified against future compatibility with OGSA and Globus 3.0. They will use the following Globus modules: GSI, GridFTP, Replica Catalogue and the Meta Data Directory to obtain attributes of data objects.

After this information obtained from each Task a first evaluation of the compatibility between WP3 software, Globus and DataGrid software have been done. This is clearly a vital evaluation with serious consequences for the success of the integration phase, testbed and prototype 0 in month 12.

In the following months each Task has to describe external interfaces which its software will provide, a list of interfaces for interoperability between modules in each Task and between them and a general view of the architecture used for each Task that will be the base for making an architecture of the whole WP3. With this information, the Integration Team will evaluate the WP3 implementation how a set of tools and services and the compatibility with Globus and DataGrid software.

Task 3.5 is following the time schedule presented in the Task 3.5 SRS document [8]. In Month 3 the definition of the software requirements for each Task was done (See SRS documents for each task). The Architecture Team together to the WP4 and WP3 Integration Teams have defined the reference platform (operating system, kernel, packages), which will be used to build, integrate, and deploy the WP3 software [12]. CVS central CrossGrid in Karlsruhe and WP's servers are available. PSNC have created such repository for WP3 in Poznan. HTTP repository is still not available, but Task 3.5 together Poznan group will provide this central HTTP server at WP3 web page in order to have a repository for the packaged source and binary distributions of the WP3 software. The final idea is to use this repository by the application Task packages (WP1) to server their application kits.

For this Month 6, Task 3.5 has established co-ordination with the other WP's and Grid projects. Concretely, we participated (April and July) in two meetings between CrossGrid WP4 and EDG Integration Teams at CERN. A local (see WP4 deliverable for Month 6) testbed is being placed with Globus and DataGrid's tools (version 1.2 beta 9) in several sites, where each Task will continue developing and testing their tools and services.

In the next Months (Month 6 – 12), implementation of the first prototype running on local grid infrastructure has to be done. Each Task will release the initial version of software to the WP3 Integration Team and will deploy this software in testbed set-up on selected site. Initial tests about performance and interoperability of this first WP3 release must be done. The Integration team will write documentation about the implementation process.