



Deliverable D1.1.1

Application description including use cases for task 1.1

WP1 CrossGrid Application Development

Document Filename: **CG-1-D1.1.1-UvA001.doc**

Work package: **WP1 CrossGrid Application Development**

Partner(s): **UvA, Univ. Linz**

Lead Partner: **UvA**

Config ID: **CG-1-D1.1.1-UvA001**

Document classification: **PUBLIC**

Abstract: This document describes the specifications for the "biomedical application" and its interaction with other components within and without the CrossGrid system



Delivery Slip

	Name	Partner	Date	Signature
From	Dick van Albada	UvA	31-May-2002	
Verified by				
Approved by				

Document Log

Version	Date	Summary of changes	Author
0.1-DRAFT_a	18-feb-2002	Draft version	Dick van Albada
0.1-DRAFT_b	12-mar-2002	Extension for Part of Univ. Linz (GVK)	Dieter Kranzlmüller
0.1-DRAFT_c	02-April-2002	Added use case; further details	Dick van Albada
0.1-DRAFT_d	01-May-2002	Corrected after the TAT review	Elena Zudilova
1.0-DRAFT	14-May-2002	Version for final internal review	Dick van Albada
1.0 -FINAL	31-May-2002	Final deliverable	Dick van Albada

CONTENTS

1. INTRODUCTION	4
1.1. PURPOSE	4
1.2. SCOPE	4
1.3. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS	5
1.4. REFERENCES	5
1.5. OVERVIEW	6
2. OVERALL DESCRIPTION.....	7
2.1. PRODUCT PERSPECTIVE	7
2.1.1. <i>System interfaces</i>	8
2.1.2. <i>User interfaces</i>	9
2.1.3. <i>Hardware interfaces</i>	11
2.1.4. <i>Software interfaces</i>	12
2.1.5. <i>Communications interfaces</i>	13
2.1.6. <i>Memory constraints</i>	14
2.1.7. <i>Operations</i>	15
2.1.8. <i>Site adaptation requirements</i>	18
2.2. PRODUCT FUNCTIONS.....	18
2.3. USER CHARACTERISTICS	18
2.4. CONSTRAINTS	18
2.5. ASSUMPTIONS AND DEPENDENCIES	19
2.6. APPORTIONING OF REQUIREMENTS	19
3. SPECIFIC REQUIREMENTS.....	20
3.1. EXTERNAL INTERFACES	20
3.2. FUNCTIONS	20
3.3. PERFORMANCE REQUIREMENTS	20
3.4. LOGICAL DATABASE REQUIREMENTS	20
3.5. DESIGN CONSTRAINTS.....	20
3.6. STANDARDS COMPLIANCE.....	20
3.7. SOFTWARE SYSTEM ATTRIBUTES	20
4. APPENDIXES.....	21
4.1.1. <i>Appendix A - Use Case</i>	22
4.1.2. <i>Appendix B - Questions from TAT</i>	24
4.1.3. <i>Appendix C - Questions to WP1 by WP2</i>	26
4.1.4. <i>Appendix D - Questions to WP1 by WP3</i>	29
5. INDEX	31

1. INTRODUCTION

1.1. PURPOSE

In this document we aim to describe the specifications for the “Biomedical application” of Task 1.1. One of the important outcomes of this description will be the requirements as regards:

1. Internal interfaces
2. Other CrossGrid Workpackages
3. External resources, such as HLA and Globus

1.2. SCOPE

In Task 1.1 we propose to develop a Grid-based prototype system for treatment planning in vascular interventional and surgical procedures through near real-time interactive simulation of vascular structure and flow. The system will consist of a distributed near real-time simulation environment, in which a user interacts in Virtual Reality (VR) and other interactive display environments. A 3D model of the arteries, derived using medical imaging techniques, will serve as input to a simulation environment for blood flow calculations. The results will be presented in a specially designed virtual reality environment. The user will be allowed to change the structure of the arteries, thus mimicking an interventional or surgical procedure. The effects of this adaptation will be analysed in near real-time and the results will be presented to the user in the virtual environment. The work in this task is embedded in the research on medical applications at the UvA and will be performed in close collaboration with the Leiden University Medical Centre (LUMC).

The software will consist of several components, that may be executed at different locations on different hardware platforms.

The principal components (not all of these are to be developed within CrossGrid) are:

1. Medical scanners (data acquisition systems – from outside CG).
2. Segmentation software for medical 3D images (LUMC, outside CG).
3. Database with scanned medical images and meta data (interface with CG)
4. Blood flow simulator with interaction capability (based on LBM, largely developed outside CG; to be integrated)
5. History DB (to be developed).
6. Visualisation software, suitable for several interactive 3D visualisation platforms.
7. Interactive measurement facilities for the analysis and quantification of simulation results with an interactive display environment.
8. Interaction module, allowing the user to adjust simulation and display parameters
9. Interface modules, to facilitate coupling of visualisation, interaction and simulation, and to regulate the data traffic between the various modules.

The Grid Visualization Kernel (GVK) will address the problem of interconnecting distributed simulation sources with visualisation clients. The idea of the GVK is to provide a middleware layer extension for scientific visualisation, which allows interactive, near real-time visualisation of running grid applications on arbitrary visualisation devices. The GVK consists of three distinct modules:

1. An input interface for delivering the simulation data to the GVK.
2. An output interface for delivering the visualisation data from GVK to the output device.
3. The GVK itself, which connects the simulation and the visualisation via their respective interfaces.

Of particular interest for GVK is the performance of the network resources. GVK will provide sophisticated compression and abstraction mechanisms, depending on the available network throughput.

1.3. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

APART	Automatic Performance Analysis: Resources and Tools ESPRIT IV WG
CAVE	Cave Automatic Virtual Environment
CAVELib	CAVE Library – API for building virtual environment (configuring display devices, processes synchronisation, creation of stereoscopic views, etc.)
CAVERN	A client/server network transfer facility intended to be used in distributed VR applications such as CAVE
CrossGrid	The EU CrossGrid Project IST-2001-32243
DataGrid	The EU DataGrid Project <project no.>
DIVERSE	Device Independent Virtual Environments (reconfigurable, scalable, extensible)
G-OCM	Grid-enabled OMIS-Compliant Monitor
G-PM	Grid-enabled Performance Measurement tool
GVK	Grid Visualization Kernel
ISS	Interactive Simulation System
HLA	High Level Architecture (DOD)
LBM	Lattice Boltzmann method
MPI	Message passing interface
OMIS	On-line Monitoring Interface Specification
OpenGL	Open Graphics Library
OpenDX	Open Visualization Data Explorer
PATOP	Performance analysis tool
PVM	Parallel Virtual Machine
R-GMA	DataGrid relational Grid monitoring architecture
Segmentation	The process of dividing (an image) into regions with similar properties or content (e.g. blood, bone, connective tissue, etc.)
SQL	Structured query language
VE	Virtual environment
ViaVoice	Speech recognition software (IBM)
VL (VLAM)	Virtual Laboratory AMsterdam – a collaborative analysis environment for applied experimental science
Volumizer	A high-level, immediate-mode volume rendering application programming interface (API)
VR	Virtual reality
VR Juggler	A scalable and extensible platform for virtual reality applications.
VTK	Visualization Toolkit

1.4. REFERENCES

APART	www.fz-juelich.de/apart
CAVE	C.Cruz-Neira, D.J.Sandin, T.A.DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In SIG-GRAPH'93 Computer Graphics Conference, pp. 135-142
CAVELib	http://www.vrco.com/CAVE_USER/
CAVERN	http://www.openchannelsoftware.org/projects/CAVERNsoft_G2/
CrossGrid	CrossGrid Project Technical Annex CROSSGRIDANNEX1_V0.1.DOC

DataGrid	DataGrid Project Technical Annex <document no.>
DIVERSE	http://www.diverse.vt.edu/
G-OCM	CrossGrid CG-3.3.1-TEM-0003-SRSTemplate
GVK	http://www.gup.uni-linz.ac.at/crossgrid/gvk
ISS	also called DEE for Dynamic Exploration Environment (and many other names as well). See e.g. R.G. Belleman and P.M.A. Sloot: The Design of Dynamic Exploration Environments for Computational Steering Simulations, in M. Bubak; J. Mościnski and M. Noga, editors, Proceedings of the SGI Users' Conference 2000, pp. 57-74. Academic Computer Centre CYFRONET AGH, Krakow, Poland, October 2000. ISBN 83-902363-9-7, and Z. Zhao; R.G. Belleman; G.D. van Albada and P.M.A. Sloot: Scenario Switches and State Updates in an Agent-based Solution to Constructing Interactive Simulation Systems, in Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), pp. 3-10. January 2002.
HLA	https://www.dmsomilpublic/transition/hla/
MPI	Available public domain – ftp://info.mcs.anl.gov in pub/mpi
LBM	Many references. See e.g. our recent publications: A.G. Hoekstra and P.M.A. Sloot: Wall effects on density fluctuations in the GBL thermal lattice gas automaton, Computer Physics Communications, vol. 142, nr 1-3 pp. 151-154. December 2001 and B.D. Kandhai; A. Koponen; A.G. Hoekstra and P.M.A. Sloot: Iterative momentum relaxation for fast lattice-Boltzmann simulations, Future Generation Computer Systems, vol. 18, nr 1 pp. 89-96. September 2001. ISSN 0167-739X.
OMIS	www.bode.informatik.tu-muenchen.de/~omis
OpenDX	http://www.opendx.org/
OpenGL	http://www.opengl.org/ ; http://www.sgi.com R-GMA
PATOP	www.bode.informatik.tu-muenchen.de/~patop
PVM	Available public domain – ftp://cs.utk.edu in pub/xnetlib
SQL	http://www.sql.org/
ViaVoice	http://www-4.ibm.com/software/speech/dev/
VL (VLAM)	http://www.dutchgrid.nl/VLAM-G
Volumizer	http://www.sgi.com/software/volumizer/
VR Juggler	http://www.vrjuggler.org/
VTK	http://www.kitware.com/vtk/

1.5. OVERVIEW

The remainder of this document is structured as follows:

Chapter 2 contains a description of the application, its various components and the manner in which they are supposed to interact. In this chapter, section 2.1.7 on Operations is central.

Chapter 3 addresses (further) specific requirements. As most of these have been extensively treated in chapter 2, they are not repeated here. Section 3.1 enumerates the requirements on the external interfaces.

Appendix A provides a description of a use case, outlining how the system is to be used.

The other appendices provide answers to specific questions asked by the Technical Architecture Team, and WP 2 and 3.

2. OVERALL DESCRIPTION

2.1. PRODUCT PERSPECTIVE

The application to be developed in task 1.1 will be a prototype of a component of a medical system supporting the diagnosis, treatment planning and patient following for arteriovascular diseases. The role of the component to be developed in task 1.1 is to support the treatment planning process by providing an interactiveimmersive environment for studying the predicted effects of proposed interventional procedures.

Each of the components of the application in Task 1.1 has its own specific requirements, besides the requirements for producing an integrated system. Therefore, the following subsections will be divided into sub-sub sections specifying these requirements.

2.1.1. System interfaces

Application to grid computational resources. The biomedical application will need access to shared grid resources for the blood flow simulation engine.

User to grid, to application, non-interactive (through portal). The user should have the possibility to start simulations and to monitor and control their progress through his portal. The information about progress has to be delivered automatically, i.e. via the SMS message about their completion. A user also should have the possibility to reserve a 3D-visualisation environment on the grid and to preview the simulation results. For the purposes of this document, the portal will be assumed to comprise that part of the user interface that can be accessed from a normal PC, or even a PDA.

Medical image DB to simulator. The simulator needs to access the medical image DB to obtain initial model information and boundary conditions. Selected results need to be stored back to the image DB.

User to application, through immersive interaction system. In addition to the functionality offered by the portal, the user should also be able to control the visualisation system in detail, edit (in a limited way) model grids for the simulator (retrieved from the image DB, and made available in a suitable format for the simulator), control the storing of annotated information to the DB.

Here a user can apply small modifications to the proposed bypass structure that should allow a fast convergence of the blood-flow simulation. Simulations of the resulting changes in the blood flow should be initiated immediately, so that the results will be available as early as possible. The progress of the simulation and the estimated time of convergence should be available for inspection through the (integrated) portal. The external interfaces to WP2 and WP3 are specified in the section 3.1 of the document.

Simulation to visualisation. In this interface, two modes of operation can be distinguished. While the simulation is running independently, the results must be stored in the History DB for later transfer to the selected visualisation system. This can be done at the simulation site, the visualisation site, or a third location.

When the simulation is coupled directly to the visualisation, selected data must be transferred to the visualisation system, with the option of storing some results for later retrieval.

The GVK will play an important role in the manipulation and transfer of the image data between the simulator and the visualisation system.

The initial goal of the GVK is to support tasks of WP1 and, first of all, biomedical simulation and flood crisis simulation. But if requested the support can be also provided to the other WPs as well. The GVK will be tested within the application of Task 1.1 of WP1. Additional interfaces to other WPs are unnecessary at the moment. More information on the GVK can be found in the second part of Appendix B “Questions about the GVK”.

2.1.2. User interfaces

The user interface to the pretreatment vascular reconstruction system will be provided in two different modes - graphical and speech recognition. By means of the following interface a user will have access to:

- Medical image database;
- History DB;
- Simulator;
- Visualisation system;
- Interactive measurement system.

The graphical part of the interface provides the visualisation of the simulated results using SGI's Volumizer (for volume rendering on systems that support this software), Vtk and OpenGL libraries (for surface rendering). The interaction system includes a voice recognition module.

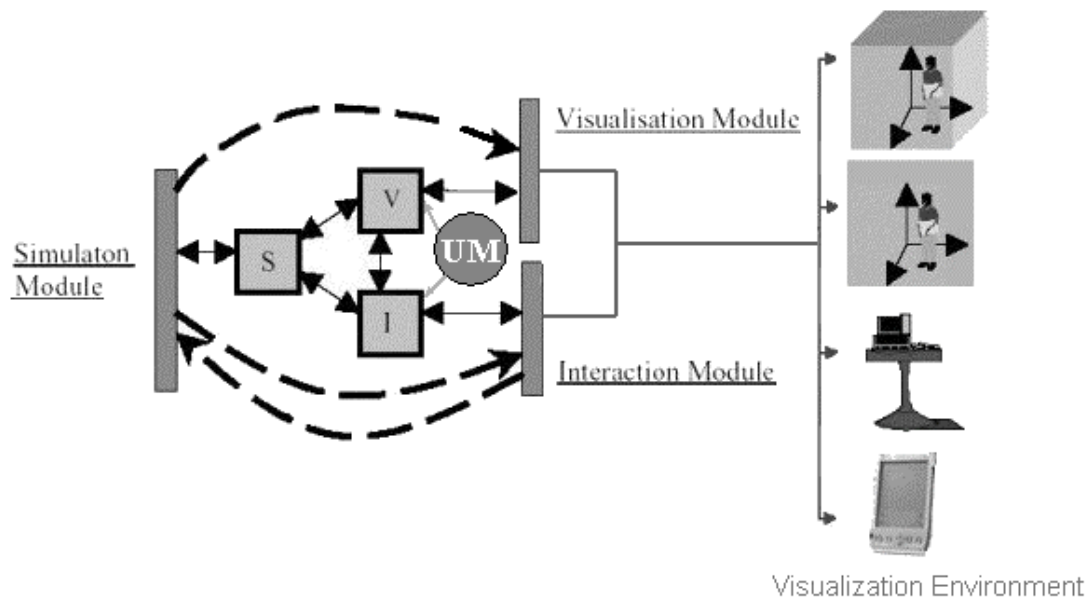


Fig. 1 The scheme of interaction within adaptive simulation-visualisation environment. UM marks the User Model described in the text.

Different display techniques will be investigated. The interaction and visualisation parts will be run in stereoscopic 3D immersive virtual environments (CAVE or the *Distributed Real-time Interactive Virtual Environment* developed at the University of Amsterdam: UvA/DRIVE) as well as on workstations and PDAs providing 2D interaction and a projected representation of 3D objects (see Fig. 1).

It is planned to apply the approach of adaptive interface design based on user modelling. Fig. 1 represents the possible architecture that the vascular reconstruction system will have if the methodology of interface adaptation will be applied. Based on the User Model, the adaptive user interface identifies each user on the base of special criteria and generates a customised simulation-visualisation environment for each individual.

The function of an adaptive interface is to customise the system environment in accordance with the human factors that characterise a person as the user of the vascular reconstruction system. The main idea that we have is to provide the vascular reconstruction system with an adaptive user interface that will permit users to manipulate both simulation and visualisation parameters with a minimal effort and in accordance with their experience and

preferences. Based on the User Model this interface will generate the most comfortable variant of human-computer interaction for each individual.

Today we are not sure what elements of the interaction will be made adaptive and what human factors will be included to the User Model of the system. Interaction capabilities and functionality will be investigated to match. This complicated task can be solved only in collaboration with future users of the vascular pretreatment system in future. For these case studies we collaborate with the Leiden University Medical Center (LUMC).

It is planned to compose a set of dialog scenarios during the interaction with several surgeons. When scenarios are formed, they will be analysed. As a result, the general elements of interaction of the highest importance for surgeons will be selected. It is planned to investigate the possibility to adapt each of these interaction elements in accordance with the experience of surgeons, their every-day tasks, preferences and habits.

The appropriate information about users will be gathered by a test component of the pretreatment vascular reconstruction system based on the User Model. The real-time identification of users will permit to change the customised environment even during the exploration process. Our final goal is to provide each user of the pretreatment vascular reconstruction system with the most comfortable environment from available alternatives. It is also planned to investigate the various projection modalities that best fit the user's tasks and personal features.

As for the GVK compound, a user can differentiate between two types of interaction:

- local interaction within the visualisation device;
- interaction affecting the simulated data.

GVK will only be accessed via its interfaces. The processing of the data will be transparently performed by GVK in the background. No adaptive features is planned to be applied here.

2.1.3. Hardware interfaces

All modules will interface with the underlying computing and communication hardware through a (standard) operating system, such as Linux, Solaris, IRIX or a version of Windows. This will be taken for granted.

Medical image database

Will be coupled to MRI/CT scanners.

Simulator

No special hardware (High performance parallel computing system)

History DB

No special hardware

Visualisation system, Interactive measurement system and Interaction system

The visualisation system, interaction system and interactive measurement system will presumably share a single VE equipped with various interaction devices, including a 6 degrees-of-freedom (DOF) hand-held "wand" and a voice recognition system.

Interface modules

No special hardware

2.1.4. Software interfaces

Medical image database

Globus / VL

Simulator

MPI; HLA / CAVERN

History DB

TBD.

Visualisation system, Interactive measurement system

OpenGL / Performer / CAVElib (/ VR Juggler) / OpenDX / VTK

Interaction system

ViaVoice / DIVERSE

Interface modules

HLA / Globus / VL

It is not necessary to support any special library on the testbed. Except for VL (VLAM) that has been developed by the UvA, all the software enumerated above is available at no cost or is commercial software purchased by the UvA. This software is either being currently in use or planned to be used for the internal deployment of the Task 1.1.

The short information about each of this software is given in section 1.3. Links and references are collected in section 1.4. Information on communication software can be found in section 2.1.5 of the document.

2.1.5. Communications interfaces

Medical image database to simulator - this interface will be used only a few times (possibly once only) during a session to obtain the initial data and later to store simulation results. The amount of data transferred will be in the order of 100 MB. Transfer time should be less than 2 minutes. Globus, GridFTP, VL

Simulator to visualisation system - This interface will carry the highest traffic. Simulation results will be transferred to the visualisation system every few seconds. The volume of this data traffic will depend heavily on attainable optimisations in the visualisation, on user behaviour and on the performance of the simulator. It will exceed 10 MB per second sustained over periods up to a few minutes.

It is necessary to find the best solution for interconnecting the simulation engine and the visualisation module of the vascular reconstruction system. Two very different alternatives – HLA and a low-level interface, like CAVERN are under consideration.

The open source software *CAVERN* is a low-level library designed for use in distributed VR applications such as CAVE. It provides set of wrappers for sockets, threads and mutexes with simple interfaces. There are also a number of helpful classes built on top of sockets wrapper for file transfers, UDP and TCP packet routing (reflectors and multicasting), RPC etc. *CAVERN* has embedded multi-socket features, which means that multiple sockets can be used simultaneously for a single connection. The first version of *CAVERN* (G1) was built on Nexus, which is part of Globus, so that version was compatible. As for the current G2 version, it does not require Globus any more, but it can be built to use Globus.

HLA (the High Level Architecture) is a general-purpose architecture for simulation reuse and interoperability. HLA aims to establish a common architecture for simulation to facilitate interoperability among simulations and promote the reuse of simulations and their components. On the base of the license agreement (currently available to the UvA) HLA provides a robust architecture with which distributed discrete event and other types of simulations can be designed. As a successor to the DIS (Distributed Interactive Simulation) protocol, it provides significant benefits to the modelling and simulation community in terms of increased software reusability, reduction in overall development and maintenance costs, and the ability to link live players and operational systems with simulations. Whether it is possible to integrate HLA with Globus is not yet quite clear at the moment; this is being analysed.

Simulator to history DB

The history DB will be used to store simulation results awaiting visualisation and to make user-initiated rollbacks possible.

Interaction system to visualisation system

Internal to VE / GVK (Optimised network transportation is most crucial for GVK, therefore Globus-IO library will to be used as a relatively low level interface to the Grid.)

Interactive measurement system to visualisation system

Internal to VE

Interaction system to simulator

Presumably HLA

Interaction system to medical image database

VL

Interface modules

HLA / GVK

2.1.6. Memory constraints

Medical image database

Disk: Average scan is $512^2 * 128 * 2$ bytes = 64 MB. Average simulation result: ~100 MB

Simulator and visualisation system

16 node Beowulf cluster for LB-simulation

Core: ~1GB (including GVK)

Data transfer

Typical data rate :~ 20 MB/s

History DB

Rough estimate of the storage requirements for pre-computed results:

10 % * 32 M pixels * 24 bytes per frame: 100 MB

60 frames per systole: 6GB

10 models per patient 60GB

2.1.7. Operations

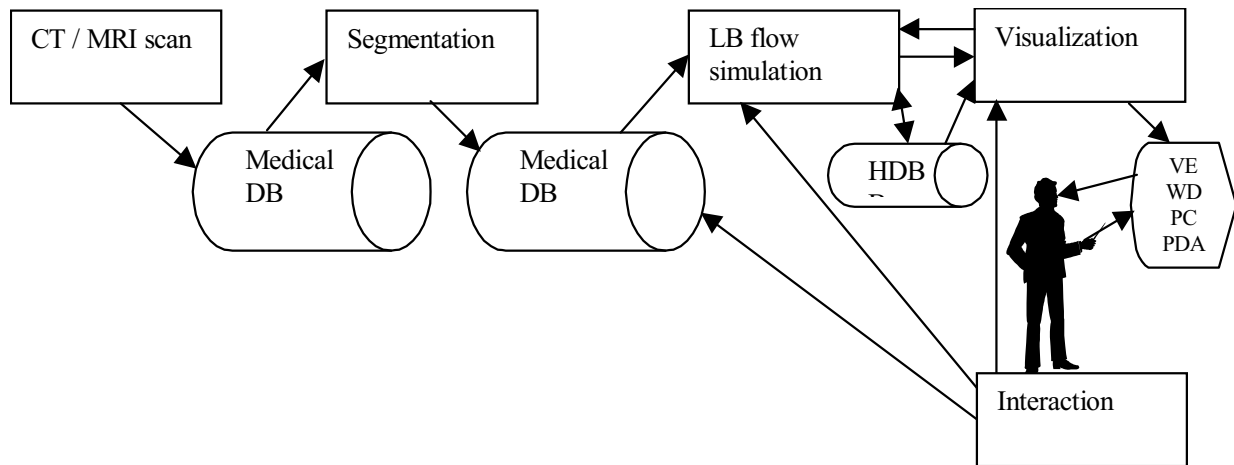


Fig. 2 The main components of the system for treatment planning in vascular interventional and surgical procedure. The portal through which non-interactive tasks can be monitored and controlled, and through which resources are reserved, is not shown.

The pretreatment system of vascular reconstruction system consists of the following main functional compounds (see Fig. 2):

- **CT** (computed tomography)/**MRI** (magnetic resonance imaging) scanners that produce the angiogram of a patient.
- **Segmentation** and image processing routines are used to generate LBM grid from raw medical scans. 3D grid editor allows a user to add and/or remove interactively areas in LBM grid according to the procedure that is simulated.
- **Medical image databases** (not within scope of CG-project). Medical databases that we are going to connect within the vascular reconstruction system are mock-up databases. They will be located at the UvA site because of security reasons. The direct access to real medical databases used in hospitals can not be provided because of privacy regulations.
- **Simulation environment (LBM)**
The simulation environment is aimed to show the effect of a planned surgical procedure. As the aim of the procedure is to improve the blood flow to the selected area, surgeon must have some means to compare the flow before and after the planned procedure. A simulation environment is used to accomplish this task. It calculates pressure, velocity and shear stress of blood flowing through the artery. The principal parameters of the selected bypass structures and the specific are stored in the patient **DB**.

A user can interact with the simulation environment (see Table 1).

Command	System response
Start simulation	Simulation is signalled to begin execution.
Stop simulation	Simulation is signalled to exit.
Pause simulation	Simulation is signalled to pause.
Continue simulation	Paused simulation is signalled to continue.

Table 1. List of simulation commands

A LB-simulation kernel will be run on shared grid resources. The state-save and rollback capabilities will be added using the history DB (**HDB**), as well as on-the-fly modifications to the computational grid. It is also planned to support the collaborative work with more than one simulation at the same time. In this case rollback capabilities are also becoming very important

– **Visualisation system**

The visualisation system is able to visualise CT and MRI data in stereoscopic 3D. But its main goal is to present the results of the simulations while the system allows inspection and probing (qualitatively and quantitatively) of the simulation results. So it provides means to perform measurements, annotate observations, inspect the flow of the blood stream, etc. The intermediate representation between the internal data structures in LB and the visualisation is an object, which is a structured grid containing a vector field (the flow velocity vectors) and a scalar field (either pressure or shear stress).

The main commands for the user interaction with the visualisation system can be found in Tables 2,3.

Command	System response
Move/translate <direction> <n> meters/feet/foot/inches	Move an object in the specified direction. <direction> may be left, right, down, up, front, forward, back or backward. <n> may be any number.
Rotate <axis> <n> degrees	Rotate an object around the specified axis. <axis> may be x, y or z. <n> may be any number.
Set scale/size <n>	Change the size of object to either an absolute value, or relative to its current size.
Copy	Copy selected object to clipboard.
Cut, delete, remove, erase, destroy, kill	Delete an object.
Make bigger/smaller, scale down/up	Increase or reduce the size of an object by 20%.
Wireframe, solid, surface, points, transparent, translucent, see-through, opaque, not transparent	Change the representation of an object.
Come here	Move the object to the hand location.
Make/set/paint <colour>	Change the colour of an object. <colour> may be one of white, black, green, blue, red, purple, yellow, orange or cyan.
Hide/show object	Hide an object from view (doesn't delete it) or show it again.
Link/unlink	Form or break a link between a selected object and the object currently in focus. Subsequent command on one object of a link will also be applied to the other objects.

Table 2. List of visualisation “actor” commands.

These commands can be applied only when a graphical primitive (object) is selected or in focus.

Command	System response
Increase/decrease sample rate	Increase or decrease the sample rate at which to extract an isosurface for a contour.
Increase/decrease contour	Increase or decrease the contour value at which to extract an isosurface contour.

Set contour to <n>	Set the contour value at which to extract an isosurface to the specified value.
Value range	Get the value range of the volume.

Table 3. List of visualisation “volume” commands.

These commands are recognized only when a volume is selected or in focus.

GVK is a grid middleware extension for interactive visualisation. Within CrossGrid, the main application area of GVK will be the visualisation of 2D and 3D data grids and triangle meshes. It will address the problems of distributed visualisation on a set of heterogeneous devices. GVK receives the simulation data from a simulation server and processes the data according to the given visualisation request (isosurface extraction, vector field visualisation, etc.). The calculated model is multicasted over the grid to the desired visualisation devices that visualise the received models.

2D and 3D grid data is considered at the moment as input for volume visualisation and isosurface generation. However triangle meshes will be supported as well. As the GVK is directly connected to a visualisation device, there is no explicit output data, however this data strongly depends on the type of the utilised visualisation device, the visualisation requested and the available network bandwidth.

The interconnection between the simulation and the GVK is currently implemented by a proprietary interface which reads the visualisation data from memory and sends it to GVK. In the future GVK should be extended to provide interfaces towards visualisation toolkits such as OpenDX and VTK.

- **Interaction system** provides the user with the interface to both simulation and visualisation modules of the system. In addition to the user commands mentioned above several common interaction commands exist (see Table 4).

Command	System response
Hello	Says "hello" (Test function to see whether the voice recognition is on).
Create <object>	Create an object, says "<object> created", <object> may be one of sphere, cube, cone, cylinder, box, spline or bypass.
Paste	Paste object from clipboard.
Select <object>	Select an object, when the object was found.
Deselect, unselect	Unselect an object.
Again, repeat	Repeat the last executed voice command.
Help	Gives a list of all objects currently in the environment.
Hide menu, close menu, open menu, show menu	Show/hide menu.
Show shadow, hide shadow	Show/hide shadows.
Quit program, exit program	Exit from the vascular reconstruction system

Table 3. List of common interaction commands

For more information, please, refer to the Appendix A “Use Case”. See also presentations on Task 1.1.

2.1.8. Site adaptation requirements

TBD

2.2. PRODUCT FUNCTIONS

The system is a near real-time interactive simulation-visualisation complex that helps surgeons in pre-surgical treatment and vascular reconstruction. The interactive environment being developed gives a surgeon the possibility to verify whether the treatment that he is going to use is appropriate in the current circumstances. The system puts a surgeon into an experimental cycle simulated by a computer and let him apply his expertise to find the right solution among the existing variants. A simulation compound of the complex simulates the parameters of a patient's blood flow, i.e. velocity, pressure and shear stress on a parallel supercomputer. A visualisation compound of the complex presents the graphical interpretation of simulated results together with some antropometric data about patient obtained from MRI or CT scanners.

GVK is a grid middleware extension for interactive visualisation. It can be used with its own interface or alternatively with traditional visualisation toolkits. It is planned that GVK will also provide optimisations for visualisation (LOD, occlusion-culling, reference rendering).

2.3. USER CHARACTERISTICS

The intended end-users of the system will be medical doctors (radiologists, vascular surgeons, i.e. domain experts without specific computer science expertise), with some support from technical experts.

Although GVK will be developed mainly for the medical personnel in mind, its architecture should be as flexible as possible to address other visualisation needs of CG (and other grid applications as well).

2.4. CONSTRAINTS

- a) Regulatory policies – no immediate constraints within CrossGrid. Anticipate on regulations regarding patient information.
- b) Hardware limitations (e.g., signal timing requirements)
Human in the loop
- c) Interfaces to other applications
- d) Parallel operation
The various modules in the application will run in parallel; the simulator is a parallel program in its own right, using MPI as the internal communication package.
The visualisation module similarly is distributed and parallel
- e) Audit functions – non known
- f) Control functions
- g) Higher-order language requirements – C, C++, Prolog*, possibly other languages
* - Amzi Prolog (www.amzi.com) is currently in use. The reason why it has been selected is because it can provide a simple interface for reasoning and knowledge representation. Thus the scenario control knowledge is represented by text, that can be easily processed by Prolog. Amzi Prolog provides a user-friendly interface for developing Prolog programs.
- h) Reliability requirements
The eventual application will be used by expensive domain experts; any unreliability will result in high cost and low acceptance.
- i) Criticality of the application
The application will be used in a medical domain. All the results should be physically correct. Special tests are going to be developed for testing this correctness.
- j) Safety and security considerations
As the application will work with patient data; security will eventually be an important issue. It is planned to encrypt the transmitted data whether it is necessary. At the moment the grid security infrastructure is being explored. In the case if HLA will be selected as a solution for building the interface between simulation and visualisation modules of biomedical application the HLA Federation Security Process have to be supported.

The detailed information is available on <https://www.dms0.mil/public/library/projects/hla/guidelines/fsp1.2-final.pdf>.

- k) Network throughput and latency

2.5. ASSUMPTIONS AND DEPENDENCIES

The application depends on components of the Globus toolkit, and on the results of other CrossGrid workpackages, as described in section 3.1 This dependency partly dictates the apportioning of the system's requirements.

2.6. APPORTIONING OF REQUIREMENTS

In the first year, the emphasis will be put on fulfilling the requirements concerning the user-interaction with the system. This is largely independent of the grid, but grid related aspects like latency hiding and, of course, grid enabling will play a central role. In year 2 and 3 actual use of the grid and integration with the tools provided by WP 2 and 3, as well as a further development of the application will take place.

3. SPECIFIC REQUIREMENTS

3.1. EXTERNAL INTERFACES

The following tools provided by WP2 will be helpful.

- 1) Software for MPI code debugging and verification for LB-Simulator
- 2) Metrics and benchmarks tools for LB-Simulator
- 3) Tools for performance evaluation and prediction for experiment configuration and GVK

More information can be found in Appendix C "Questions to WP1 from WP2".

From WP3.1 a portal for the user interface is necessary for the Interaction system. It will be used to configure an experiment as a whole. The user should have the possibility to monitor the progress of the simulation through this portal. He also will be able to use the advance information for the reservation of the visualisation environment if necessary. A wide range of projection devices (modalities) from VE to desktop PCs and PDAs should be accessible (see section 2.1.2).

From WP3.2 resource management is needed for optimisation of LB-simulation and processing power and storage capabilities of the grid.

From WP3.3 monitoring will be helpful in configuring experiments and analysing the network load and bandwidth during visualisation.

More information can be found in Appendix A "Use Case" and Appendix D "Questions to WP1 from WP3"

3.2. FUNCTIONS

TBD

3.3. PERFORMANCE REQUIREMENTS

Performance requirements are implicitly described in section 2.1.7 (operations) and in Appendix A (Use Case)

3.4. LOGICAL DATABASE REQUIREMENTS

Logical database requirements are implicitly described in section 2.1.7 (operations) and in Appendix A (Use Case)

3.5. DESIGN CONSTRAINTS

No additional constraints known.

3.6. STANDARDS COMPLIANCE

3.7. SOFTWARE SYSTEM ATTRIBUTES

There are a number of attributes of software that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified. A partial list of examples can include: reliability, availability, security, maintainability and portability.

4. APPENDIXES

4.1.1. Appendix A - Use Case

Assumption: The system is used for pre-operative planning for the abdominal aorta.

Step 1

An MRI scan ("Angiogram") is obtained for the patient. This is a 3 D image with a typical resolution of about 1mm^3 and about $512 * 512 * 128$ pixels. A 3D-visualisation system is reserved for use in step 3. In the near future higher resolutions are expected.

Step 2

The image is segmented so that a clear picture of the important blood vessels and the location of aneurisms and blockages is obtained. These features occupy some 10% of the original image.

Step 3

Using the segmented image, a computational grid for a LB simulation is generated

A simulation of the normal pulsatile blood flow in the vessels is started. Required input from the physician: parameters like pressure drop (possibly time dependent)

Run time: several hours on a fast 16 node Beowulf cluster. (enter Globus)

Step 4

Using an interactive 3D-visualisation system, the physician studies the vascular structure (and if step 3 completed, the flow through the vascular structure) and proposes several (3 to 5) bypass designs. These are drawn in the 3D image, using a library of standard bypass designs, and are used to generate alternative computational grids.

Estimated duration of step 4: order of 1 hour.

Step 5

The blood flow simulations for the bypasses are initialised using the new grids and the (partially) converged results from step 3. Each of these simulations will require several hours on a fast 16 node Beowulf cluster to suppress the start-up transients. (enter Globus)

Step 6

The physician can monitor the progress of the simulations through his portal. He will be informed automatically, e.g. through an SMS message of their completion. The physician can use the advance information to reserve a 3D-visualisation environment for step 7.

Step 7

The results of the simulation are presented in the 3D-visualisation system, either from the stored history of one heartbeat (systole), or from the running simulation. Here the physician can apply small modifications to the proposed bypass structure that should still allow a fast convergence of the blood-flow simulation (like small changes in the angles, and/or position where they are connected to the existing blood vessels).

Simulations of the resulting changes in the blood flow should be initiated immediately, so that the results will be available as early as possible. The progress of the simulation and the estimated time of convergence should be available for inspection.

The principal parameters of the selected bypass structures, plus the comments of the physician are stored to the patient DB.

Estimated duration of step 7: order of 1 hour.

Step 7 constitutes the core of task 1.1. Here a "human-in-the-loop" simulation is performed and the requirements on interaction and immediate access to simulation power are the most severe.

Steps (4), 5, 6, 7 may be repeated, but this should be avoided as much as possible.

Estimated time scale of the whole procedure: one day (with sufficient access to HPC/HTC resources).

Time scales within step 7:

Typical rates:

- Screen refresh: > 25 Hz.
- Interaction, response to user movement: >~10 Hz
- Interaction response ~ 1sec

Timings

1. Response in 3D visualisation to head movements: < 0.1 second
2. Response to change in user focus (display at high resolution of different parts of the image)
3. Response to user input like
 - Rotation/enlargement of the image < 0.1 sec
 - Colour coding < 0.1 sec
 - Display of a different aspect of the flow (pressure, flow velocity, shear stress) 1 sec -5sec
4. Grid editing (drawing): < 0.1 sec
5. Play time for one systole - under user control from 2 sec (from stored data) to 2 minutes
6. Time for simulation results of a small modification to become available: 5 minutes.
This is speculative and definitely cannot be realised for high-resolution simulations with large filling factors. A requirement like this will result in a trade-off in the attainable accuracy and will affect the computational strategy.
For a sufficient accuracy, it may be necessary to repeat steps 5 through 7, where the first step 7 provides a preview, after which high-resolution computations are started, the results of which are presented in the second step 7.

Data transfer requirements.

Data transfer requirements are most severe when simulation results need to be presented directly in an interactive simulation in step 7, or when the local storage at the display station is insufficient to store all pre-computed simulation results. Typical data rate here are in the order of 20 MB every second.

Rough estimate of the storage requirements for pre-computed results:

10 % * 32 M pixels * 24 bytes per frame: 100 MB

60 frames per systole: 6GB

10 models per patient 60GB

4.1.2. Appendix B - Questions from TAT

Answers (partial) to questions from TAT to Task 1.1 in CGArchitectureForWP1.ppt:

Answers are tentative and should not (yet) be taken to constitute a definitive "specification".

- How to build connection between simulation kernels and GVK for interactive use? (possible solutions HLA, VISIT, Cavern)?
A layered architecture will be used. In this respect various functionalities must be distinguished: setting up a the system, setting up a connection within the system, exchanging control information, exchanging bulk data.
Details of this layering still have to be studied before a definitive answer can be given.
- Is it going to be one of the resources or rather Grid service build over Globus?
That will partially depend on the nature of the libraries used. HLA presumably will be a "required resource". GVK will adopt the functions Globus for internal operation and transportation (to be studied in more detail).
- The two tasks have many in common (GVK, interactive simulation) – how to unify common parts? (possible solutions CCA)
Presumably tasks 1.1 and 1.2. This will require further examination of both tasks. It will be necessary to study the visualisation requirements and take the different visualisation approaches into account during design of GVK.
- Interactive Distributed Data Access - not applicable to task 1.1
- MPI standard and fault tolerance
Not really an issue, but as the task will require frequent restart checkpoints anyway, a solution to this problem should not be too hard (location nature of the checkpoint dumps is critical, of course).
- Requirements to
User Interface (portal). A portal should eventually be helpful in configuring the experiment as a whole, but obviously will not constitute the core of the user interface. When and where used, it should be accessible using a wide range of devices/modalities, from VE to desktop machines and PDAs
WP 2 tools.
MPI code debugging and verification. The LB code (the MPI-based) module is already fully operational. The tool may be helpful, but is not vital.
Metrics and benchmarks. These may be helpful eventually, when LB simulations will be "farmed out" to the grid. The tools should stress the responsivity and throughput of the system.
Performance tool - should be helpful in configuring the experiments.
The performance tool will be of particular interest for GVK, since the transport of the visualisation data must be adapted to the actual network characteristics.
WP3.2 resource management. These may be helpful eventually, when LB simulations will be "farmed out" to the grid. The tools should stress the responsivity of the system. Some of the optimisations in GVK may benefit from additional processing power and storage capabilities of the grid.
WP3.3 monitoring. should be helpful in configuring the experiments and analysing the network load and bandwidth during visualisation.
WP3.4 optimisation of data access. N/A.

Questions about the GVK

- Main functionality

The main application area of GVK will be the visualisation of 2D and 3D data grids and triangle meshes. GVK receives the simulation data from a simulation server and processes the data according to the given visualisation request (isosurface extraction, vector field visualisation, etc.). The calculated model is multicasted over the grid to the desired visualisation devices that visualise the received models.

- What kind of information will receive and export?

2D and 3D grid data is considered at the moment as input for volume visualisation and iso surface generation. However triangle meshes will be supported as well. As the GVK is directly connected to a visualisation device, there is no explicit output data, however this data strongly depends on the type of the utilised visualisation device, the visualisation requested and the available network bandwidth.

- Plans for integration with globus and ogsa

Optimised network transportation is most crucial, therefore globus-io library is to be used as a relatively low level interface to the grid. As for ogsa, it does not seem to be mature enough to be incorporated with GVK so far. Maybe it will be considered in future versions.

- Will the GVK be connected with the Grid Information Service? How?

Connecting the GVK to the Grid Information Service does not seem to be necessary at the moment as the user needs to specify the data source and the visualisation device anyway. Automatic scheduling is not a feasible way in this context. However the GVK will be able to check whether the requested visualisation device is available using Grid Information Services. For this purpose GIIS will be deployed.

- What are the plans for testing GVK? How will be obtained several VE distributed over the testbed and qualified people to use it at the same time?

The GVK is planned to be tested with pretreatment vascular reconstruction system. Apart various dummy data will be deployed. The visualisation can be multicasted to several visualisation devices, interaction with the simulation is restricted to one distinguished user, though.

- What will be the connection between GVK and portals

There will be no connections between GVK and portals, because GVK is a grid middleware running as background process. The user will access it through an application requesting a visualisation.

- Is there any request for resource reservation for the GVK? ie, QoS besides reducing frames per second?

QoS concerning the available is very interesting for GVK visualisation. Hence there will be a possibility to specify the required network bandwidth within the visualisation request.

- How will be the GVK affected by evolution of other grid technologies?

GVK is relatively loosely connected to the grid middleware, however GVK would benefit from improvements within the following grid technologies: QoS concerning network bandwidth; possibly resource reservation.

- Will you give support to other WPs or other grid projects for GVK?

The initial goal of the GVK is to support tasks of WP1. But if requested the support can be also provided to the other WPs as well.

- Security issues

Concerning security issues the grid security infrastructure is being exploring. Furthermore it is possible to encrypt the transmitted data.

4.1.3. Appendix C - Questions to WP1 by WP2

Answers are tentative and should not (yet) be taken to constitute a definitive "specification".

- Categories of applications

Task 1.1: interactive & near-real time, visualisation

- Which programming languages do you use?

Task 1.1: C, C++, Prolog, possibly other languages

- Do you use MPI?

Task 1.1: yes, Lattice Boltzmann module

- Which MPI calls do you use / are important?

Task 1.1:

- MPI_Barrier
- MPI_Bcast
- MPI_Cart_create
- MPI_Cart_rank
- MPI_Cart_shift
- MPI_Comm_rank
- MPI_Comm_size
- MPI_Finalize
- MPI_Init
- MPI_Pack
- MPI_Recv
- MPI_Reduce
- MPI_Send
- MPI_Sendrecv
- MPI_Unpack
- MPI_Wtime

- Will you use other / additional communication mechanisms? If so, which?

Task 1.1: MPI, CAVERN and/or HLA

- Are you using the HLA?

Task 1.1: yes, for now.

- What is the role of CCA? Will it be used in addition to MPI?

Task 1.1: undecided

- Component structure of application?

- Distribution of components?
- When / how are the components started?
- Some components will be internally parallel. Will they be executed in a distributed way (vs. on a single parallel computer)?

Task 1.1: The various modules in the application will run in parallel; the simulator is parallel in its own right, using MPI for internal communication. The visualisation module is similarly distributed and parallel.

Simulation, data storage and visualisation may be at different sites; multiple simulations can run concurrently at different locations, but each simulation will be confined to a single cluster.

- Granularity of performance analysis?

- Should it be able to 'look' inside components or is it sufficient to consider only the interactions between them?

Task 1.1: The Lattice-Boltzmann code can be a performance bottleneck, but probably does not offer many opportunities for optimisations. The data pipeline from simulation to visualisation is another critical component, where performance analysis can be important. This pipeline contains many steps, some within the modules (data selection, compression, possibly encryption, etc.) and other in the routing between modules.

- Which kinds of performance problems do you envision?

- What are the performance critical aspects of the application?

Task 1.1: data throughput, network throughput, network latency

- Which data do you need? (Classification from the APART working group)

- A) What? (Metrics)
 - Time (computation, communication, I/O) - yes
 - Data volume (communication, I/O) - yes
 - Hardware counters -
 - instructions, FLOPs, memory read/write, cache misses - could be
- B) Where? (Resolution in space)
 - Location
 - Host, node, process, thread - yes
 - Program structure - yes
 - Program, module/file, function, loop/block, statement - function/loop
- C) When? (Resolution in time)
 - Complete execution
 - Program phases (time / logical phases) -
 - Single event

Task 1.1: network performance

- Use of WP2 tools

- A) MPI code debugging and verification
- B) Metrics and benchmarks
- C) Performance Evaluation and Prediction

Task 1.1: A) The Lattice Boltzmann code (the MPI-based) module is already fully operational. The tool may be helpful but is not vital.

B) These may be helpful eventually, when LB simulations will be "farmed out" on the Grid. The tools should stress the responsiveness and throughput of the system.

C) Should be helpful in configuring the experiments. The performance tool will be of particular interest for GVK, since the transport of the visualisation data must be adapted to the actual network characteristics.

4.1.4. Appendix D - Questions to WP1 by WP3

Answers are tentative and should not (yet) be taken to constitute a definitive "specification".

3.1. Portals and roaming access

- What do the "Grid users" really need? (what functionality should the Migrating Desktop have?; which resources and applications should be available via desktop?)
Task 1.1 needs eventually:
 - Control and monitoring for farmed-out simulations,
 - Access to patient DB (i.e. also security, authorisation, authentication, etc.)
 - Reservation of unique resources (medical scanners, 3D visualisation environments)
- How can task 3.1 interact with WP1 applications?

3.2. Resource management

- The resource management is very dependent and limited on that approach Do we assume the set of resource is changing dynamically?
Task 1.1:
 - year 1 - assume pre-allocated resources and clusters
 - year 2 - assume small, static pool of clusters with background load
 - year 3 - assume larger pool of clusters (virtual organisation) with dynamically varying load and availability.

This section contains currently only a list of questions that have to be answered by or in conjunction with responsibilities of other WPs. Some questions are indirectly related to the basic services provided by the middleware developed in the DataGrid project.

- How many users have to be simultaneously supported?
Task 1.1 - only a single user
- Is there a common queue of jobs for all users or a different queue for each user?
Task 1.1 - N/A
- Which mechanisms would be available for a given user to specify different priorities between his own jobs?
Task 1.1 - A mechanism should be attractive to have. Should not require much knowledge of scheduling; should allow dynamic adjustments.
- In the case that multiple users share common services and a common queue, are we going to support priorities between users? Which would be the scope of these priorities (earlier execution, pre-emption of already running jobs with less priority, ...)? This question is closely related to the architecture of the Resource Broker (RB) developed at the Datagrid project and their implementation of the resource selection processes and the maintenance of a queue with submitted jobs.
Task 1.1 - N/A
- What parallel programming libraries are going to be used? Only MPI? Others? Which?
Task 1.1 - HLA contender for inter-module communication, not for intra-module.
- What platforms (in terms of operating system and processor architecture) are we supposed to support? What versions of GNU tools or any other required development software should we use?
Task 1.1 - mostly Linux on x86

- Are there any special requirements for WP1 applications that need to be included in the Job Description Language?

TBD

- What is understood by interactive applications running on the grid? Do they require an immediate response once they are submitted? What is going to happen if there are not enough resources to satisfy the requirements of one application at a given instant of time? Should it be simply cancelled, or postponed, or pre-empt non interactive applications already running ,...?

Task 1.1 - not all subtasks require an interactive response. For those that do, a solution must be found. One option could be: prescheduling in a specified time-window; interactive response once it runs. This may include prescheduling resources that can be freed immediately when an additional interactive task is submitted. When there are insufficient resources for a running task (e.g. resources are reclaimed after all), migration of the task is the preferred solution. Other options still need to be examined.

- Which are the performance indexes that should be typically optimised: total execution time, response time, resource utilisation efficiency,... In the latter case, is there going to be any mechanism to decrease priorities of users who have consumed large number of resources in the past? This question deals also with the existence of accounting mechanisms.

Task 1.1. At least two modes (see use case): Non-interactive: minimise total execution time; Interactive: minimise user response time for "focussed tasks"; focus may change dynamically.

- What is the expected size of our grid environments and the applications to be run there (in terms of number of processors or number of processes, respectively)? Are there any estimations of the scalability requirements?

TBD

- Are there any restrictions in the use of external packages? Are we restricted to the use of software with a GPL license and source code publicly accessible? Let's give an example. DataGrid uses Condor-G as part of their Job Submission Service. Condor-G is only distributed in binary form under an academical license or an internal use license. What are the implications to CrossGrid if Scheduling Agents are partially based on the services of DataGrid's Job Submission Service? Is this completely acceptable or are there any limitations?

TBD

3.4 Data optimisation

See use case

5. INDEX