The IST Programme
Project No. IST-2001-37153

GEMSS

Grid-enabled Medical Simulation Services

http://www.gemss.de

Deliverable D1.3b
Evaluation & Validation

Status:	Final Release
Version:	1.1
Security:	Public
Responsible:	MPI
Authoring Partners:	all

Release History

Version	*Date*	
0.1	29/07/04	initial draft
0.2	12/08/04	1st revision
0.3	19/11/04	2nd revision
0.4	02/12/04	3rd revision
0.5	16/12/04	4th revision
0.6	20/12/04	5th revision
0.9	21/12/04	release candidate
1.0	22/12/04	Final release
1.1	31/01/05	Status update

Executive Summary

This report constitutes Deliverable D1.3b of the GEMSS project. This version contains the final validation and evaluation report of the GEMSS system.

The GEMSS infrastructure consists of a generic service provision framework, a hosting environment, a certificate authority, one or more service registries, and a pluggable client side component framework.

This final evaluation and validation report includes:

- an evaluation of the final GEMSS system,
- a survey regarding the application developer experience, and
- a test of the performance of the GEMSS system.

The results of the evaluation and validation can be summarised as follows:

The final version of the GEMSS middleware meets all central goals defined in the initial requirements specification. Middleware requirements that were not fully implemented are of low priority (only relevant in the exploitation phase).

All six medical applications have been successfully Grid-enabled and turned into Grid services using the final version of the GEMSS middleware.

Stress tests were conducted to burden the middleware with a high load sufficient to break it. These tests revealed an initial problem within the external MySQL software used in the server side parts of the GEMSS framework (failure of a locking mechanism). After fixing this locking problem, performance and robustness tests validated the middleware design and demonstrated the high quality of the infrastructure.

A long term test of the middleware provided information about the standard use patterns and the efficiency and high reliability of the final GEMSS middleware. Negotiation and file transfer overheads were measured and found acceptable for the GEMSS applications. Test showed that the middleware is well suited for applications with typical run-times > 10 minutes such as the six GEMSS pilot applications.

Table of Contents

List of Tables

List of Figures

2 Introduction

This report constitutes Deliverable D1.3b of the GEMSS project. This version contains the final validation and evaluation report of the GEMSS system.

The GEMSS architecture uses a client/server topology employing a service-oriented architecture. It is based on web service technology that allows integration with various hosting platforms. It consists of a plug-in module client framework. The module code can, thus, be released separately, allowing open-source or closed-source distribution depending on the preferences of the responsible partner.

The purpose of the evaluation and validation subtask is to:

- define the testing methodology,
- Identify metrics for evaluation,
- evaluate the GEMSS system with respect to the requirements specified in the Deliverable D1.1,
- evaluate a series of test beds, and
- test the GEMSS system performance.

This final evaluation report includes:

- an evaluation of the final GEMSS system,
- a survey regarding application developer experience, and
- a test of the performance of the GEMSS system.

Three different groups are identified each of which has a different perspective on the GEMSS system:

Group	***View***
middleware developers	system design, suitability for the intended task, stability of the system, performance
application developers	ease of installation and use of the GEMSS system, completeness of system functionality and documentation, performance, security issues
end users	ease of use, application performance, security issues, costs.

In this deliverable the focus of the analysis is on the areas that are interesting to middleware and application developers.

2.1 On Computational Grids

Increasingly, companies are realising that they can achieve significant cost savings by outsourcing non-essential elements of their IT environment to various forms of service providers. In addition, computing addresses collaboration, data and cycle sharing, and other modes of interaction that involve distributed resources. This development generates new pressure for distributed application development and deployment.

Despite the diversity of resources that arise from the continuing distribution and decentralisation of software, hardware and human resources it is essential that a desired quality of service (QoS) is maintained. This requires new abstractions and concepts that let applications access and share resources and services across distributed, wide area networks, while providing common security semantics, coordinated fail-over, or other QoS metrics that are of importance in a particular context.

Work within the community of large-scale scientific research has led to the development of *Grid technologies,* which have been widely adopted in scientific and technical computing. Grid technologies support the sharing and coordinated use of diverse resources, especially the coordinated use of high performance computing (HPC) resources.

2.1.1 A classification of Grid Applications

Grids are designed to solve scientific problems at unprecedented computational scales with complex, large and, possibly, distributed input data. Based on memory requirements and data coupling, the arising applications can be classified as different types (Snavely et al., 2003) :

1. **Loosely coupled**. Applications in this class are made up of tasks with low memory requirements and a small amount of data for each task, and little communication required between tasks. This class of application is computationally intensive and is suitable for execution on a wide area cluster connected with a low bandwidth/high latency network.
2. **Pipelined**. Here, digital streaming and/or real time data has to be processed. Often, these applications are very memory intensive, and have a coarse grained inter-task communication, while the constituent tasks are highly parallel.
3. **Tightly synchronised**. Applications in this class have frequent inter-task synchronisation. They may also have significant computation and memory/data usage. Applications of this class have traditionally been run on tightly coupled high-performance computing (HPC) Systems . They can certainly make use of Grid information services to locate the most appropriate HPC resource. However, it still has to be seen whether *truly* distributed versions of these applications with a sufficient latency tolerance to be run on wide area networks will evolve.
4. **Widely distributed**. Applications in this class search, update, and/or unify distributed databases. These typically have small requirements regarding computational power and data storage, but they need to seamlessly communicate across the Grid environment to access data bases that are variously owned and updated.

In GEMSS the Grid is mainly used to locate an appropriate HPC resource, transfer data, and execute the application in a three tier client/server environment, where the (local) client supervises the computation executed at the (remote) HPC site by utilising the middleware. Therefore, the GEMSS system can easily be used to run loosely coupled and tightly synchronised applications.

2.1.2 Benchmarks for Grid software

Historically, benchmarks for "traditional" microprocessors and HPC systems fall into at least two categories: Low level benchmark tests that determine the rates at which the system can perform fundamental operations, and representative applications that are meant to capture the computational need of a class of applications. These benchmarks were used to assess all features of a computer architecture. Usually, both categories should be represented in a suite of Grid benchmarks.

Grid benchmarks may measure the rate at which computation can be carried out (see e.g. Frumkin and Van der Wijngaart. 2002), and the rate at which data can be accessed through a deep and wide data storage hierarchy. Traditional HPC benchmarks focused on ascertaining the cost of memory access, communication, and the issue rate of functional units. However, beyond hardware differences, a critical difference between tightly coupled systems and Grid systems is the *software service layer* or *middleware* (Snavely et al., 2003).

This report is about the evaluation of the GEMSS middleware. Therefore, the focus of benchmarking will be laid on the overhead that is introduced by the Grid aspect of the applications,

including *data transfer rates, negotiation overhead,* as well as the *robustness* of the GEMSS middleware.

2.2 Evaluation strategy

In order to evaluate the implementation of the GEMSS system a number of different strategies will be pursued as outlined in the following:

Implementation status

The implementation of the GEMSS system will be checked against the requirement statements specified in Deliverable 1.1:

- Surveys will be used to capture application developers experience with the middleware, such as ease of installation, documentation of interfaces, or modularity of the implementation.
- The feature set of the final system will be cross referenced with the requirements specification.
- Achievements will be compared to the results of the midterm evaluation (D1.3a)

Stress testing the middleware

Stress testing the middleware by requesting and running a great number of jobs is designed to break the middleware in order to find weak spots in its processing pipeline. These stress tests are suited to exposing bugs and they can also be used to measure the maximal job throughput.

Performance testing

A number of performance tests were conducted to measure the overheads introduced by some of the features in the GEMSS middleware. Both the security and negotiation overheads were measured, and robustness testing under normal conditions was measured.

Long term testing

The long term system stability will be tested by running the system over a long period. A large number of jobs comprising all six example applications will be run with varying input data and parameters in order to simulate a normal load on the middleware. For each allocated job success, negotiation time, up- and download time and speed will be recorded in order to acquire statistics that characterise the robustness of the system and highlight the processing overhead introduced by the GEMSS system.

Bug Tracking

Finally, the evaluation phase will also be used to improve the GEMSS system software by fixing bugs and removing performance bottle necks. To capture and track bugs a bug tracking system has be established (GEMSS BugTrack).

3 Analysis of the GEMSS software

In this section, an analysis of the current status of the GEMSS middleware and the six medical applications will be given. First, a comparison of the original requirements specifications is cross-referenced with the design specification and the final implementation status of the middleware in order to capture the implementation status. Based on a survey conducted amongst users and the application developer ease of installation and porting of applications is rated. Then, the final status of the medical applications is documented. Finally, a set of tests is run to capture different aspects of the middleware when in action. These tests include a stress test, to capture weak spots in the processing pipeline and long term tests to analyse the robustness of the middleware under normal load. Measurements are given that describe the processing overhead introduced by the middleware. Finally, a comparison of the GEMSS middleware with other Grid efforts concludes this section.

3.1 Middleware implementation status

3.1.1 Requirements described in D1.1 v1.2

The following requirements are reproduced from the *Requirements Specification* deliverable D1.1.

req-b1: The GEMSS infrastructure must support a flexible approach to handling business models, which will allow other types of model to be added in the future. GEMSS will initially implement a telephone business model and a variable pricing model. As the project progresses this will be reviewed in light of knowledge gained during the project, allowing other models to be undertaken if required.

req-b2: Business models must support pay per use. This allows clients to pay for the time they actually use as opposed at say a fixed fee.

req-b3: Business models must support a reservation facility, which allows clients to reserve a service with an application service provider in future. This will allow clients to plan ahead, for example booking patients in for treatment.

req-b5: A verifiable audit trail is required for all business transactions. The audit trail should form legally admissible evidence should there be a disagreement over billing. Both the client and server must keep separate audit trails, which will match unless an error occurs.

req-b6: All billing must be transparent, with each clients bill clearly broken down into the costs specified in the contract of use.

req-b7: Each GEMSS service provider must support an audit trail interface, which application providers can use to download the current audit trail for their applications. For the GEMSS test-bed at least a generic licence manager will be installed, for proof of concept purposes, allowing the test-bed to record application usage and bill clients for services rendered.

req-b12: All GEMSS software components must be clearly identified and the owner/supporter specified (if any). The likelihood of support after exploitation should also be clearly indicated, to allow any potential exploitation plan to be clear over future weaknesses regarding critical but unsupported software components. Each GEMSS infrastructure component must all be owned or supported by at least one of the project partners during the project, and be covered by exploitation plans for its onward maintenance.

req-a1: Each application's user interface should be easy to use by non-technical medical users.

req-a2: Each application's user interface should provide a fast response time, avoiding interface delays that can occur with network delays etc.

req-a3: Each application's user interface should hide, as much as possible, the complexity involved in Grid computing.

req-a4: A well defined mechanism must be provided for service discovery. This will allow clients to discover which application service providers are available, as in a directory lookup service. The providers will still need to be asked if they have cycles free for specific jobs.

req-a5: A well defined mechanism must be provided for service description. This will allow clients to discover which services an application service provider supports.

req-a6: A mechanism for automated quality of service negotiation must be supported. A minimal requirement is for semantic descriptions to be available for business models, which allow automated negotiation for computation jobs within a larger contractually agreed business model.

req-a7: Provision must be made for the monitoring of jobs in progress.

req-a9: A well defined mechanism is required for application installation to an application service provider. This installation procedure should also be simple, keeping installation costs low. This is needed to support creation of new applications, and update of existing applications with the latest software versions. It is up to the application providers, however, to write their software to conform to the installation procedure requirements, including splitting the application into client and service components.

req-a10: A well defined mechanism is required for submitting a client request to an application service provider.

req-a11: A well defined mechanism is required for delivering results to a client.

req-a12: The developed GEMSS infrastructure must be open source. However, it should be possible to distribute the infrastructure along with commercially sensitive and therefore closed-source applications or enhancements. This implies that the licensing model should be something like the LGPL rather than the GPL open source license.

req-a13: The developed GEMSS infrastructure must be modular in nature, with well-defined interfaces between each module to allows for future upgrade or re-writing of specific modules.

req-a14: The GEMSS infrastructure must be easy to install, maintain and administer. This is an essential requirement for exploitation.

req-a15: The GEMSS infrastructure must be easy and inexpensive to install and use. Third-party software components that require large licence fees must be avoided. Proprietary partner software cannot be used if it will be unavailable after the project, unless alternatives are available and supported, allowing for a realistic exploitation model at the end of the GEMSS project.

req-a20: All services running on a remote application service provider must have an appropriate software licence. This may involve negotiation with application vendors to provide a Grid capable licence system. A flexible approach to licensing is required.

req-a21: Application service providers must support LINUX, UNIX and/or Windows applications in order to run the medical services. The GEMSS test-bed should provide at least one site that supports each of the LINUX, UNIX and Windows operating systems, in order to successful demonstrate each medical service.

req-a23: The GEMSS infrastructure should be robust, containing at least some error recovery mechanisms such as redundancy and network re-direction in case of service failure. Robustness must be considered in infrastructure design.

req-a25: Accounts data should be easily viewed by authorized personnel.

req-p1: Quality of service will be measured according to the listed terms.

req-p2: A high level of system stability is required, which can generally inspire client confidence.

req-p3: The infrastructure should be capable of specifying an optional "time of completion". The consequences of not completing a job by this time should be specified in the business contract penalty clause.

req-p4: The GEMSS infrastructure must be capable of handling files up to 1.5Gbytes in size, and computational timeframes ranging from minutes to hours. Specifically it should handle the listed performance criteria.

req-s1: The security infrastructure must support user authentication.

req-s3: Standard PKI encryption, at the highest reasonable level for patient data, of all transmitted confidential data is required.

req-s4: All patient data must be anonymised where possible. In the case of head image scans, or other inherently personal data, this cannot be done and data should always be treated as personal patient data.

req-s5: Commercial data and business models are also confidential in nature and must be encrypted for transmission and protected while at service provider sites.

req-s6: Partner firewalls must be tunnelled through in ways acceptable to partner site system administrators.

req-s7: Confidential data, particularly patient data, must only be stored at the server site for the period of computation. Once the results have been downloaded all copies of the data at the server site must be removed.

req-s8: Both client and server sites must instigate a formally declared logging procedure for security reasons. This will include access logging, and should allow security breaches to be detected and appropriate action taken.

req-s9: The perceived main security threat, for all clinical partners, is from internal staff.

req-s10: There must be no remote shell access over the Grid. Shell access is a major security weakness. Currently the only partner system that required shell access is the COPHIT system, and alternative methods for the COPHIT interface will be investigated.

req-s11: The number of account types supported by GEMSS should be minimised, to help eliminate security risks associated with authorisation administration. So far, three types of account have been identified: GEMSS users, developers and administrators.

req-s14: The GEMSS infrastructure must itself be constructed with security in mind. It should be possible to formally document the GEMSS infrastructure security features, so that client and third-party organizations can clearly see why GEMSS is secure. Best practice in security should be followed in the implementation and operation of GEMSS.

req-s16: State of the art security measures should be reviewed, work package two deals with this, and features such as intrusion detection, resistance to denial of service attacks and access controls specified. Appropriate measures will be built into the GEMSS security infrastructure.

req-s17: GEMSS should comply full with the IETF X.509 standards, the most widely used standard for the definition and use of digital certificates.

req-l3: Each health practitioner, and all their functions, and each technical resource should be precisely identified and located. A generic plan of the network could be very useful.

req-l4: The European Directive on the Medical Devices should be taken into account.

req-l5: It is very important to define clearly and exhaustively all flows of personal data involved when the GEMSS applications are in use because of the ruling of these processing of personal data

by the applicable national privacy law transposing Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data.

req-l7: Personal data must be processed fairly and lawfully.

req-l10: Personal data must be kept in a form which permits identification of data subjects for no longer than is necessary for the purposes for which the data were collected or for which they are further processed (medical and scientific research purposes but also keeping records for liability reasons).

req-l11: The processing of personal data must comply with the criteria selected by the applicable national law for making data processing legitimate.

req-l13: The rules related to the confidentiality and the security of the processing of personal data should be enforced.

req-l15: The communication of personal data concerning health might be ruled by special regulations in some Member States (e.g. cf. *Infra* Medical secrecy rules).

req-l16: The controller must notify the competent supervisory authority before carrying out any wholly or partly automatic processing operation or set of such operations intended to serve a single purpose or several related purposes.

req-l18: Privacy of healthcare data is paramount

req-l19: Patient confidentiality must be preserved throughout the Grid solution process. Only anonymous image data should be transferred, there will be no additional healthcare information associated with image data available outside the user's clinical department.

Exploitation requirements

Exploitation requirements are not expected to be satisfied until the exploitation phase. They are included here for completeness, but will not be evaluated further.

req-b4e: Business models must provide for a level of client support, and a mechanism for paying for such support. (E)

req-b8e: A licence manager, independent of the service provider and acceptable to the application providers, will be installed as part of any exploitation plan. (E)

req-b9e: There should be a mechanism for compensation in the case of a system failure, system error or some breach of contract. This mechanism should be instigated before resorting to legal recompense and will be detailed in the contract of use. (E)

req-b10e: There must be financial reward for providing application expertise on the Grid. (E)

req-b11e: There must be financial reward for providing hardware accessed on the Grid. (E)

req-a8e: The infrastructure should be capable of being used with a variety of server batch systems. This will allow additional service providers to be integrated onto the Grid for further exploitation, and allow upgrades or changes in existing service providers computational batch systems. (E)

req-a16e: The GEMSS infrastructure must be capable of supporting various client operating systems. Support for Windows, Linux and UNIX running on Pentium and Itanium class systems are likely to be needed. (E)

req-a17e: The GEMSS Grid infrastructure should be capable of accommodating new service providers and services, once properly authorized. This should be possible without major disruption, for example re-installation of infrastructure components, to existing clients and service providers. (E)

req-a18e: Remote users should be able to check the status of jobs, potentially from anywhere in the world. (E)

req-a19e: The GEMSS infrastructure should provide options for teleworking, which involves running and managing the run of jobs remotely. (E)

req-a22e: A list of explicitly defined faults and error codes must be created for any implementation of GEMSS. This will allow business contracts to unambiguously specify which faults they can recover from, as well as response times and charges in the event of errors. (E)

req-s2e: An integration path must be created to allow use of the NHS hospital authentication system. (E)

req-s12e: All GEMSS sites, both client and server, must be responsible for their own security. It is the individual partner's responsibility to uphold their stated security procedures, and they are responsible for security breaches at their sites. (E)

req-s13e: All GEMSS sites must publish a minimal set of practical, cost-effective and verifiable security procedures and standards they agree to uphold. These will be used to verify a site is correctly implementing security best practice. Conformance with published procedures may be the basis for legal claims or defences in the event of a security breach. (E)

req-s15e: Audit software must be difficult to hack/compromise, allowing confidence in the reported audit trails at each site. Confidence in the audit trails must be sufficient to allow billing based upon them, and legal arguments to be resolved by them. (E)

req-l1e: There must be legally binding patient consent forms for each application where appropriate. (E)

req-l2e: Contracts between supplier and client must be legally binding in accordance to EU and national law. GEMSS should create some example contracts for use during test-bed evaluation. (E)

req-l6e: The controller of a telematic network must be designated. (E)

req-l8e: Personal data must be collected for specified, explicit and legitimate purposes and not further processed in a way incompatible with those purposes. Further processing of personal data for historical, statistical or scientific purposes shall not be considered as incompatible provided that applicable national law provides appropriate safeguards. (E)

req-l9e: Personal data must be accurate and, where necessary, kept up to date. Every reasonable step must be taken to ensure that data which are inaccurate or incomplete, having regard to the purposes for which they were collected or for which they are further processed, are erased or rectified. (E)

req-l12e: The controller or his representative must provide a data subject from whom data relating to himself are collected with at least some information on why this is necessary. (E)

req-l14e: If the processing of personal data is carried out on the behalf of the controller, the controller must choose a *processor* providing sufficient guarantee in respect of the technical security measures and organizational measures governing the processing to be carried out, and must ensure compliance with those measures. (E)

req-l17e: The creation of a Internal Data Protection Authority (a board aiming at the privacy protection) within the telematic network should be considered. The function of a security officer also should be considered. (E)

req-l20e: The GEMSS software, in common with other engineering software, should include a disclaimer indicating that people use the software at their own risk. It would be made clear that the onus is on the user to validate the results. (E)

req-l21e: Patient specific simulation software should meet appropriate Quality Assurance standards for safety critical systems. (E) Design features described in D1.2a v1.4

Design attributes

design-a1: client-server service-oriented architecture

design-a2: incremental design methodology, with phased delivery Client side modules

module-c1: Business component

module-c2: Client-side authorization component

module-c3: Transport and messaging component

module-c4: Security context component

module-c5: GEMSSProxy

module-c6: Component manager

module-c7: QoS negotiation module

module-c8: User interface component

module-c9: Workflow enactment component

module-c10: Certificate store, private key store [client]

module-c11: Service discovery component [client]

module-c12: Service state repository component [client] Service provider side modules

module-s1: Compute resource manager

module-s2: Conversational authorization module

module-s3: Error recovery module

module-s4: QoS management module

module-s5: Web server

module-s6: Intrusion detection system

module-s7: Certificate store, private key store [service provider]

module-s8: Service discovery component [service provider]

module-s9: Service state repository component [service provider]

module-s10: Account service

3.1.2 Cross reference with design document and final implementation status

Table 1 Captures the current implementation status with respect to the initial requirements and the design specification. Requirements that were marked as exploitation requirements are not considered. All in all 43 requirements are fully satisfied – one more then predicted during the design phase, three requirements are only partially satisfied, and – since priorities changed during the project - six requirements are implemented incomplete as predicted in the design document. These requirements target the exploitation phase but were initially not considered likewise.

Req.	*Design feature*	*Requirement satisfaction*		*Remarks*
		predicted by design specification	*final*	
req-b1	module-c1, module-s10	full	full	
req-b2	module-c1, module-s10	full	full	
req-b3	module-c7, module-s4	full	full	
req-b5	module-c1, module-c12, module-s9, module-s10	full	full	
req-b6	module-s10	full	full	
req-b7	module-s10	full	full	
req-b12	IPR register	full	full	
req-a1	module-c8, module-c5, Application interfaces	full	full	
req-a2	module-c8, Application interfaces	full	full	
req-a3	module-c5, Application interfaces	full	full	
req-a4	module-c11, module-s8	full	full	
req-a5	Service registry	full	full	
req-a6	module-c7, module-s4	full	full	
req-a7	module-s11	full	full	
req-a9	XML application description plus tools	full	full	
req-a10	module-c3,module-s5	full	full	
req-a11	module-c3,module-s5	full	full	
req-a12	LGPL release of project code	full	full	
req-a13	module-c6	full	full	
req-a14	GEMSS client release, XML application description plus tools, Tomcat	full	full	
req-a15	IPR register	full	full	
req-a20	module-c1 [licence options will be selected before QoS negotiation]	partial	partial	Licence models are agreed as part of a legal contract enabling a service provider to run the application developers software. There is no on-line negotiation of licensing, but different instances of a service could use a different licence supported by a licence manager like FlexLM.
req-a21	Use of Java	partial	partial	The client infrastructure runs under Windows and Linux. The server infrastructure has been installed only under Linux.
req-a23	Client exceptions, module-c5, module-s3	full	full	
req-a25	module-s10	full	full	
req-p1	module-s4	full	full	
req-p2	incremental design methodology	full	full	
req-p3	module-c1, module-c7, module-s4	full	full	
req-p4	module-c3, module-s5	full	full	
req-s1	module-c4, module-s2	partial	full	

Req.	*Design feature*	*Requirement satisfaction*		*Remarks*
		predicted by design specification	*final*	
req-s3	module-c10, module-s7, module-c4, GEMSS CA	full	full	
req-s4	Applications	incomplete	incomplete	The legal analysis advised against trying to anonymise images of patients. Instead we treat all data as personal data, apply best practice security, and expect well defined contracts between the data controller and processors.
req-s5	module-c4, module-c3	full	full	
req-s6	client driven web service invocation	full	full	
req-s7	module-s11	full	full	
req-s8	module-c10, module-s7, module-s6	full	full	
req-s9	[app partners need written procedures as suggested by D2.2a, ST4.3, ST4.4, ST4.5 access will be governed by the University and NHS security protocols	incomplete	incomplete	The project has contacted the NHS information services security dept and fed into their policy making. However, formal contact with health care providers has been left to exploitation.
req-s10	module-s5, firewall	full	full	
req-s11	module-s11	full	full	
req-s14	D2.2a	full	full	
req-s16	D2.2c	full	full	
req-s17	module-c4	full	full	
req-l3	[app partners need written network / personnel maps]	incomplete	incomplete	The mapping of networks and personnel can only be done on a case by case basic. The methodology is provided in the security deliverables, but the actual task can only be started in the exploitation phase once the parties involved are known.
req-l4	D2.2b	full	full	
req-l5	D2.2b	partial	partial	Clear definitions of the flow of personal data can only be mapped out during exploitation, when the parties involved are known. The privacy legal report gives advice on this together with an analysis of the European directives.
req-l7	D2.2b, D2.2d, D2.2e	full	full	
req-l10	module-s11	full	full	
req-l11	[Partners need to talk to legal representatives for national legal perspective]	incomplete	incomplete	Individual partners have not contacted national lawyers, since this would only be productive during exploitation, where a specific case could be analysed.
req-l13	D2.2b	full	full	
req-l15	[Partners need to talk to legal representatives for national legal perspective]	incomplete	incomplete	Individual partners have not contacted national lawyers, since this would only be productive during exploitation, where a specific case could be analysed.
req-l16	We use test data only	full	full	
req-l18	D2.2b	full		
req-l19	Applications	incomplete	incomplete	The legal analysis advised against trying to anonymize images of patients. Instead we treat all data as personal data and apply best practice security and expect well defined contracts between the data controller and processors.

Table 1 Cross reference between Requirements and final implementation status

3.1.3 Code audit

Several steps have been taken to ensure code quality. The GEMSS middleware was developed using Java. This promises portability over a wide variety of computer architectures and operating systems. Finally, Java exception handling makes it easy to locate sources of most types of bugs.

In addition to language specific features that improve code quality the following measures were taken:

- The GEMSS middleware is built upon a plug-in framework that supports a set of components. These components each have well defined interfaces and are, therefore, independent of each other from the implementation point of view. This modularisation makes maintaining the software easier.
- The component manager supports a mechanism to regularly check for existing updates and for the inclusion of new components. This removes some of the burden on the users with respect to checking for upgrades, and could make fast security patching a practical feature of the system.
- The software was developed in incremental steps including a phased delivery of the components. Hence, compatibility and interaction of the components could be tested at an early stage of the project and problems with software design could be avoided.
- To exchange and maintain component releases BSCW was used as a shared project area. While it provides the means to share software at intermediate release stages, the project wide implementation of a versions system, like CVS, could have improved communication between the middleware developers.

Below a module level code audit of progress to date is given. Full details can be found in deliverable D1.2b.

Client architecture

The client architecture is shown in Figure 1. The client would typically run an offline business workflow to open an account with a service provider and agree a payment mechanism. When a Grid job is required to be run the client would then open negotiations with a set of service providers for a particular application's job. The quality of service negotiation would then be run to request bids from all interested service providers who can run the clients job; this would result in a contract being agreed with a single service provider. The client would then upload the job input data, and potentially any application workflow, to the service provider where the server side application would take over.

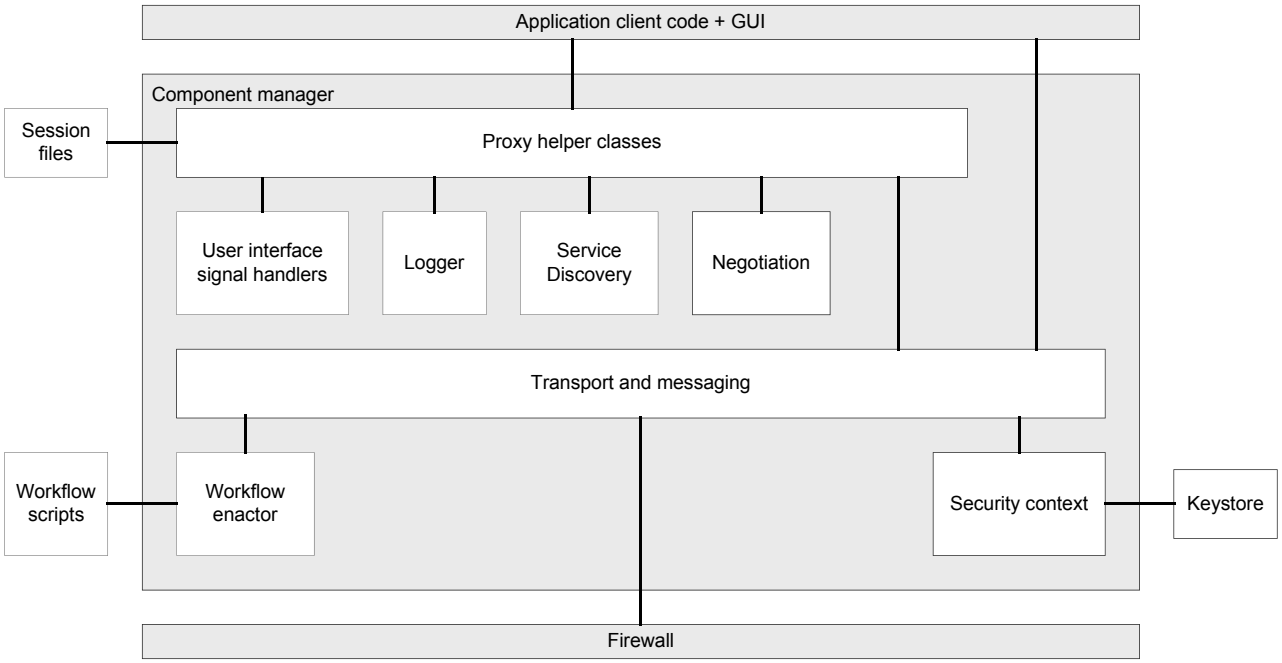

Figure 1 Client Architecture

The component manager provides a way to dynamically load different versions of components as required, allowing flexibility of implementation. All components have a well-defined interface that the component manager knows.

Most workflow within GEMSS is encoded by the proxy helper classes and supports negotiation and job handling workflow. A workflow enactor component, which can execute soft coded workflow scripts, is being investigated as a technology demonstrator.

The client always drives the three-steps of the GEMSS process. The reason for this is the requirement to operate with firewalls and not tunnel holes through them. Given this, there can be no service provider initiated connections, or call-backs, to the client.

Application client code and graphical user interface

The application code will provide a custom application user interface and manage the proxy helper classes. The major part of the client application code will be the user interface and job execution workflow.

Component manager

A plug-in framework that is capable of supporting a set of components, each with well-defined interfaces, and providing a signalling mechanism that components can use. Components in the component manager's code repository can be dynamically loaded as required by the application. The component manager will also manage client sessions and provides a start-up and shutdown function the application can call when it is first run.

The component manager supports a mechanism to regularly check for
patches to existing and completely new components. This removes some of the burden on the users checking for upgrades, and could make fast security patching a practical feature of this system. All component code is signed by the vendor, allowing for checks that code updates originate from an authorized source.

Logger

This is a low level logging component that is used by the service provider side intrusion detection system.

Negotiation

This component provides the functionality to run a client driven QoS negotiation with several service providers. GEMSS supports a basic QoS negotiation based on a request/offer model as well as an advanced negotiation model using a reverse English auction protocol that conforms the FIPA specification. The result of the negotiation process is an agreement to run a specific job following the WSLA specification.

Proxy helper classes

Proxy helper classes provide a simple interface that the applications can link to. These proxy classes are intended to be the main contact point for the applications, invoked via the component manager. There are proxy helper classes to run Grid jobs, agree quality of service details and negotiate with several service providers to get the best deal for a job. These proxies are persistent and support state, saved to session files, which allows jobs to be run over a protracted period without the need for a persistent client connection to the service provider.

Security context

This component supports maintenance of a security context. It provides a framework for generating, storing and verifying trust for security tokens of various types. It provides the security context used by the transport component for HTTPS and secure message encoding.

Service discovery

This component invokes the service registry and obtains a list of services that match a query. The registry this component searches can be specified by the client allowing only trusted service registries to be queried by the client.

Transport and messaging

Transport and messaging is a policy-based message and invocation framework. This component is capable of creating low-level messages, and sending them with appropriate transport bindings, as specified by the WSDL that defined the remote service. All calls to services are fed through the transport and messaging component. The appropriate WS security and WS policy procedures are applied before a message is sent, and before a response is passed to the other client-side modules.

User interface signal handlers

These signal handlers are a set of default handlers that applications can use. These signal handlers are only default handlers, and it is expected that a sophisticated application would create their own handlers, which would link directly to their own custom user interfaces.

Workflow enactment

This is a component capable of parsing and running a workflow language script. This component is expected to be used as a technology demonstrator for soft coded quality of service negotiation workflows.

Service provider architecture

The service provider architecture is shown in Figure 2. Applications will be run by the service under the direction and orchestration of the client, subject to the necessary availability of and authority to use resources. The compute resources available at the service provider's site will be used to actually run the services. The error recovery module is there to checkpoint services and re-starts them if required. The quality of service management module is there to handle reservations with the

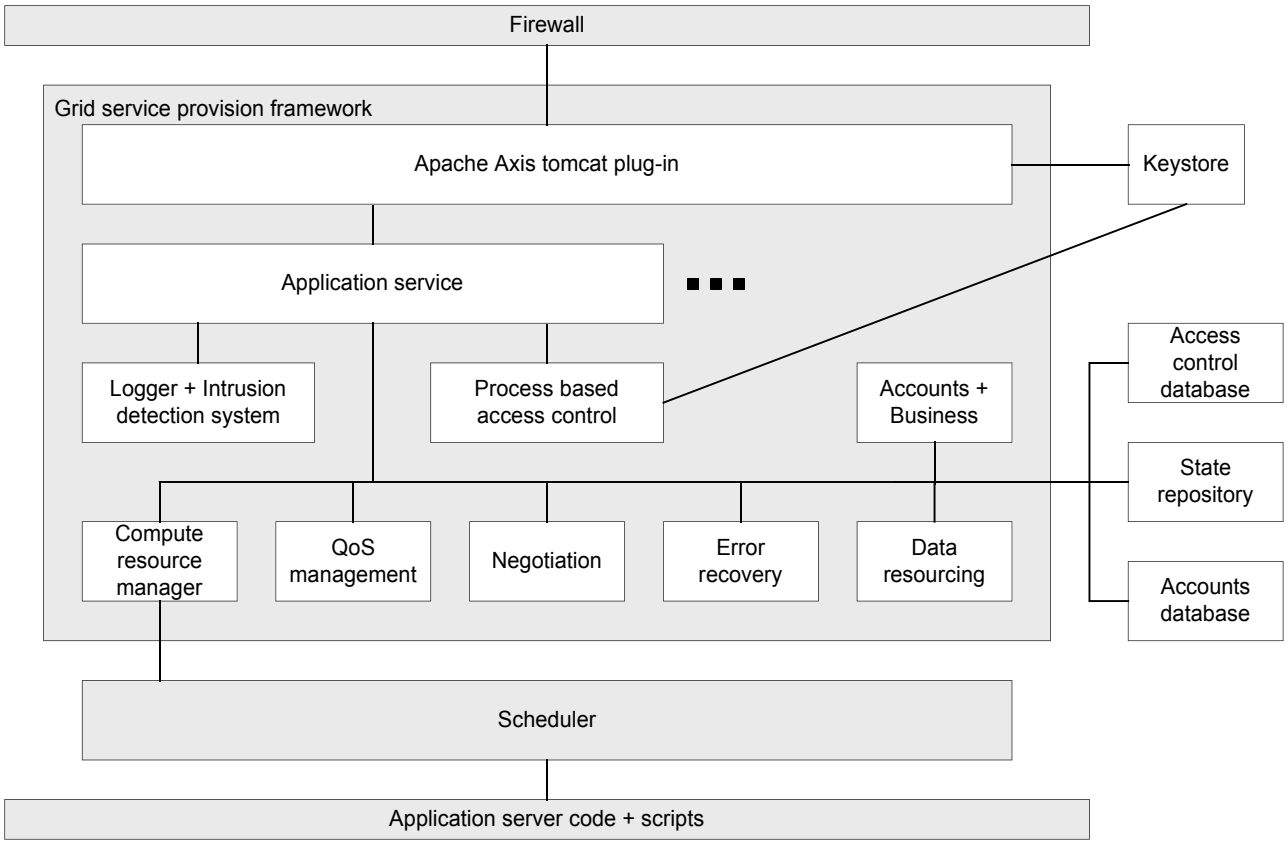

Figure 2 Service provider architecture

compute resource job scheduler and provide input to the quality of service negotiation process so that sensible bids can be made to client job requests.

Accounts and business

This service maintains a database of auditable information about client accounts, including payment details and the current billing status. When jobs are run the business modules pricing models are used to calculate the payment due for resources used. A web interface is provided to allow client budget holders to access their billing status and review their account.

Application service(s)

The application service provides all the operations defined within the service WSDL file. These operations are routed to the appropriate module by various handlers, which are optionally installed at the service provider's site. The process based access control module is always consulted before an operation request is sent to a module.

Application server code and scripts

Each application must install its server side code and a set of scripts to run these codes. These scripts all follow the same convention, and basically expect to receive an archived file containing the input data and produce a archived file with the output data. Start scripts invoke the scheduler as required by the service provider installation.

Apache Axis tomcat plug-in

Apache Axis plug-in's are used to support WS security enhancements and obtain the distinguished name of the client. Tomcat itself provides the web service interface to individual application services whose functions are exposed via a WSDL interface.

Compute resource manager

The Compute Resource Manager provides an interface to the service provider's choice of scheduler. This module is called to request and confirm resource reservations in advance, to start jobs for confirmed reservations as well as to query the status of a job. Since each service provider can have a different scheduling system along with different compute resources, there will have to be customised resource manager implementations, such as NEC's JCOSY, for each service provider, that support advance reservation capabilities.

Data resourcing

This module allows for persistent data between jobs, allowing intermediate files to be used as the input to other jobs etc. This is intended as a technology demonstrator within GEMSS.

Error recovery module

Closely linked to the compute resource manager, this module may use check-pointing to re-start or re-locate jobs as required (where the resource manager or the application itself enables this). This is intended as a technology demonstrator within GEMSS.

Grid service provision framework

The Grid service provision framework aims to hide the complexity of the Grid by supporting the task of transforming existing applications into Grid services without having to deal with the details of Web Services and Grid technologies. The transformation of medical simulation applications into Grid services is based on the concept of generic application services. A generic application service is a configurable software component which exposes a native application as a service in the GEMSS hosting environment.

Logger + Intrusion detection system

A module to analyse low level event logs with the aim of detecting intrusion. The intrusion detection module may also integrate with conventional non-grid intrusion detection subsystems associated with the web server. This is intended as a technology demonstrator within GEMSS.

Negotiation

This module provides server side support for a closed-bid reverse English auction. It supports FIPA protocols and calls the QoS management module to try to get the best reservation possible. Bids are returned to the client for interpretation by the client side negotiation component encoded as a web service level agreements (WSLA).

Process based access control

This module maintains an access control database of allowed process actions. The application service and other server side modules will check with the process based access control module before any operations are invoked. This provides a mechanism to check the clients distinguished name, obtained from the WS security header information, and confirm the operation was expected and authorized.

QoS management

The QoS management module provides an high level interface for a basic QoS negotiation utilized by the client or the negotiation module. The basic QoS negotiation includes requesting a certain QoS criteria (e.g. time, price) from a service for a specific job as well as confirmation or cancellation of the requested QoS criteria. The QoS management modules takes a QoS Request following the web service level agreement (WSLA) specification and a Request Descriptor as inputs and offers a QoS Offer (following again the WSLA specification). Internally the QoS management module utilizes the compute resource manager for resource reservations, the business module for price calculations and an application specific performance model for runtime estimations. An application specific performance model is used to estimate the execution time jobs will take given a metadata description (RequestDescriptor) of the jobs input data.

Scheduler

GEMSS assumes the existence of a service provider specific scheduling system that supports advance resource reservation. In GEMSS NEC's COSY scheduler and the MAUI scheduler are used.

Development Process

The current status of development work to date is shown in Figure 3 and 4. The blue modules have been developed, the green modules are a work in progress and the red modules are technology demonstrators. Technology demonstrator modules will not be released as part of the main software release, but will be worked on until the end of the project and form the basis of a proof of concept development. The final release of the GEMSS middleware is phase I3. There is a further release, phase I4, which has been developed until the end of the project containing the technology demonstrator work.

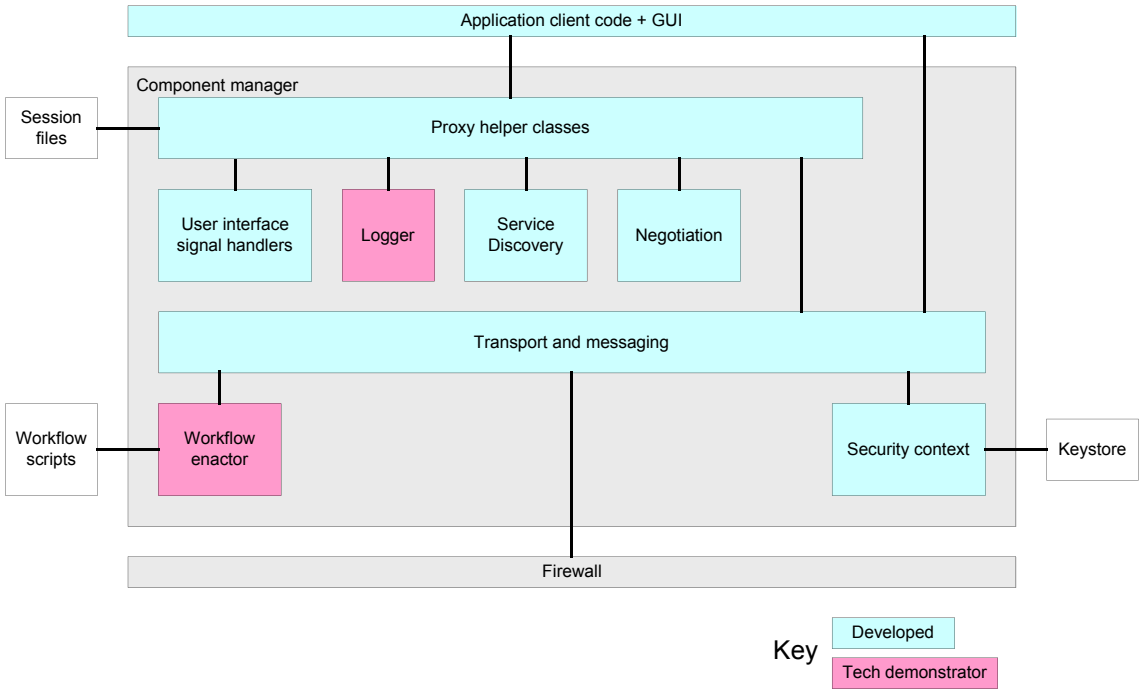

Figure 3 Client module development progress

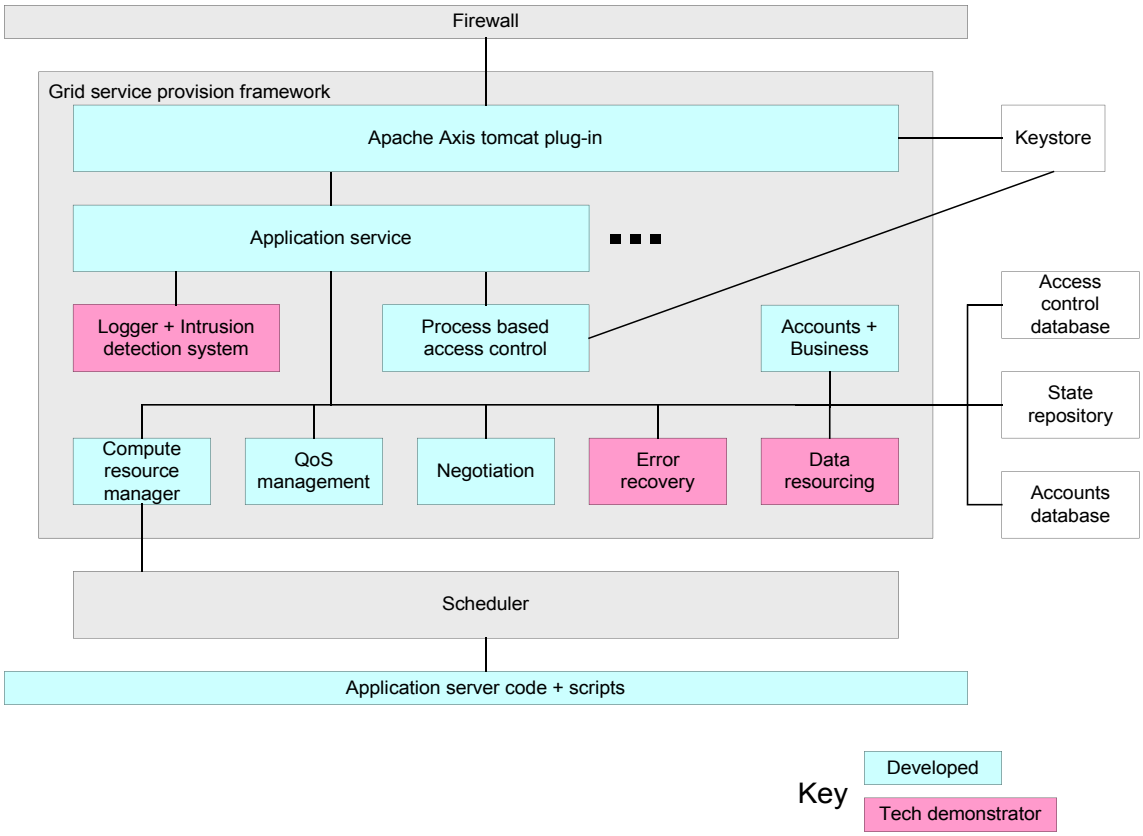

Figure 4 Service provider module development progress

3.1.4 Result of user surveys

In order to capture the middleware status regarding its documentation and its ease of installation, a survey was conducted. Application developers were asked to rate these criteria, and comment on shortcomings and possible improvements of the middleware. The results can be summarised like follows:

- Acquiring an initial keystore is very difficult. Many different pass-phrases are needed, and it is not clear why one is not sufficient.
- The initial installation requires editing of text files and it requires that the very same configuration values are entered multiple times.
- Porting an application to the middleware at a basic level is straightforward, but the more advanced features are not well documented.
- The middleware gives a lot of warnings that can irritate the user, and error messages often do not clarify whether the application or the middleware failed, and hence, whether the application developer can deal with the problem, or whether she should submit a bug report to the middleware developers.
- The service providers use different server environments (i.e. different schedulers and different command shells). Therefore, it is necessary for the application developer, to write specific scripts for each application provider site.

Naturally, the following suggestions arise to improve this situation:

- A (graphical) installation program could be added to guide the application developer and end user through the initial configuration. This would hide most of the complexity of the current text file based configuration and it would circumvent platform incompatibilities.
- An application porting tutorial would be very helpful and could promote more advanced features of the middleware.
- The readability of warnings and error messages could be improved.
- A standardisation of the server side execution environment would it make easier to deploy applications at different service provider sites.

3.2 Medical Applications Summary

For each application a short summary of its purpose and implementation status is given. Detailed reports on the applications can be found in the GEMSS deliverables D4.1, D4.2, D4.3, D4.4, D4.5 and D4.6 respectively.

3.2.1 ST 4.1 FEBiNA - Maxillo-facial Surgery Support

This application deals with pre-operative planning for maxillofacial surgery. The surgery first separates the maxilla from the rest of the skull by osteotomy and then uses distractors attached to a head mounted halo frame to exert forces on the midface and change its shape. Tools for computing the facial outcome are based on the Finite Element Method (FEM). The magnitude of the computing resources required for this application emphasizes the need for a Grid solution. GEMSS offers real potential for refinement of the head modeling process, especially the possibility to accurately model the forces by using advanced nonlinear simulations.

Fulfillment of the original requirement specifications:

req-st4.1-1: A Linux workstation is required to run the client application software. It will need a high performance graphics card to run the graphical front end software.

Comment: The client workstation should have at least 1.5GB of RAM, the more the better. The basic toolchain has only modest requirement on graphics cards. However, if additional components like landmark picking using real-time volume renderers are activated, requirements on the graphics card are higher (at least 128 MB of memory and a fragment shader).

req-st4.1-2: A mechanism to upload DICOM medical image data, originating from CT head scans, to the client workstation is required. These data transfers will be up to 200Mbytes in size.

The quality of service requirements are as follows:

Service completion time	Interactive 10-60 mins, High resolution few hours, transient/full detail overnight.
Service Throughput	Depends on service completion time (< 10 per day)
Service Security	Anonymisation, encryption, authorization, authentication.
Service Availability	>90 %

Comment: The upload of DICOM data to the client is achieved easily by LAN or CDROM.

Response time for a high-resolution model with about 400 000 elements is about 10 min for a linear simulation in total (including less than 5 min for the proper simulation on the minimum number of three dual-processor nodes and immediate cluster availability). The actual times also depend on the number of preprocessing steps involved (in particular, additional mesh and image filters necessary to reliably obtain good results). Transient runs require several hours of computation time, and tend to show much larger variations in run time.

Service throughput and availability depend on the stability of middleware and cluster. For linear simulations, throughput is certainly much higher than the requested 10 simulations per day. Availability can be improved substantially by using several servers, which is supported by the GEMSS system by implementing service discovery and service negotiation.

Service security is guaranteed by the GEMSS middleware (temporary directories are still world readable on the server during the testing phase). Anonymisation is performed manually by the user.

req-st4.1-3: The software in Table 2 is required on the client workstation.

Meshing Software VGrid	proprietary NEC
GRAL mesh library and related tools	open source NEC/proprietary NEC
OpenDX or JULIUS (with appropriate plugins) as graphical front-end	open source + proprietary plugins NEC
ITK toolkit (segmentation & registration)	open source

Table 2 ST 4.1 Client software requirement list

Comment: JULIUS software and ITK are not required (optional). A source installation of OpenDX is required, in order to compile the plugins for the bone cutting tool.

Meshing Software VGrid	proprietary NEC
Mesh and image manipulation tools	proprietary NEC
OpenDX (with appropriate plugins) as graphical front-end	open source + proprietary plugins NEC
MPI image processing software(segmentation & registration)	open source MPI

Table 3 ST4.1 Revised client software requirement list

req-st4.1-4: The software in Table 4 is required to be installed on the server.

Parallel Finite Element Code HeadFEM	proprietary NEC
FE code ADINA	Linux version, commercial license
DRAMA load-balancing library and related tools	open source NEC
PILUTS linear solver library	proprietary NEC
Scheduling system software "COSY"	proprietary NEC

Table 4 ST4.1 Server software requirement list

Comment: The ADINA software is not required. In addition to Table 4, an installation of HyPre from LLNL is required. The successor of HeadFEM is called FEBiNA, and is required instead of HeadFEM. If also model preparation is performed on the server, mesh generation and image cutting application have to be present on the server instead of the client.

Parallel Finite Element Code FEBiNA	proprietary NEC
DRAMA load-balancing library and related tools	open source NEC
PILUTS linear solver library	proprietary NEC
HyPre Boomer AMG linear solver	open source Lawrence Livermore National Lab
Scheduling system software "COSY"	proprietary NEC

Table 5 ST4.1 Revised server software requirement list

In comparison with D1.3a (mid-term evaluation), the following progress has been made:

QoS: A performance prediction model for the linear case has been implemented, which uses an analytical formula involving the number of elements of different types, the requested accuracy and the number of computational nodes, and computes the required runtime and main memory. The necessary data is extracted on the client side and converted into a GEMSS request descriptor. When a job is executed on the server, the input model is repartitioned to the required number of processes. The simulation software can work on any number of computation nodes and thus optimally exploits the server's capacities.

Preprocessing: A number of additional tools have been developed and added to the toolchain, like landmark selection, improved segmentation allowing to classify muscles, and various mesh and image filters which enhance the robustness and degree of automatisation.

Simulation: A linear model of 400 000 elements currently runs on average in about 270s on 3 nodes (6 processors), using 12 GB of memory. Non-linear simulations have proven to be much more dependent on mesh quality and certain separation conditions of moved and fixed bone parts.Mesh filters have been implemented in order to guarantee the necessary properties. Runtime for nonlinear simulations is considerably longer (in the order of several hours).

Error handling: The server side scripts now detect and catch more error conditions like abnormal termination.

3.2.2 ST 4.2 QUARTS - Quasi Real-Time Surgery Support

The major shortcoming of image-guided surgical planning based on pre-surgically acquired functional MRI (fMRI) data is the brain shift phenomenon. The occurrence of surgically induced deformations invalidates positional information about functionally relevant areas. This problem is addressed by non-linear registration of pre-operative fMR images to intra-operative MRI acquired by an Open-MR scanner, or to intra-operative 3D ultrasound data. So whenever an intra-operative dataset is acquired, the following image processing chain must be executed: (a) transfer of anatomical images from the scanner and conversion into a machine-independent data format, (b) correction of intensity non-uniformities in the scan data, (c) linear registration of the (low-resolution) intra-operative scan with a (high-resolution) pre-operative scan, (d) intensity adjustment between both scans, (e) non-linear registration of both scans to yield a 3D deformation field, (f) application of the deformation field to the pre-operative fMRI datasets, (g) overlay of the deformed functional information onto the intra-operative data, (h) conversion and transfer to the presentation device (e.g., monitor, surgical microscope).

Fulfillment of the original requirements specification

req-st4.2-1: A LINUX workstation is required to run the client application software.

Comment: Requirement fulfilled.

req-st4.2-2: A mechanism to upload up to 4 image datasets to the client workstation is required; these datasets can originate from CT, MRI, SPECT and PET scans. The data transfers will be up to 100 Mbytes in size.

The quality of service requirements are as follows:

Service completion time	10 minutes
Service Throughput	1 or 2 per day
Service Security	anonymisation, encryption, authorisation, authentication
Service Availability	>99% at reserved times when surgery is under way

Comment: Due to the unavailability of an Open-MR scanner data can only be uploaded to the client workstation via LAN or CDROM. Runtime of less than 10 minutes has been achieved for the processing stage of the tool chain for the available patient data set. Real-life runtime depends on the latency and current load of the target HPC system and the internet.

Anonymisation of the data has to be done by the user manually. Encryption, authorisation and authentication are handled by GEMSS middleware. Service availability depends on the stability of middleware, cluster hardware and communication/scheduling software.

req-st4.2-3: The software in Table 6 is currently required on the client workstation.

Visualization tool for a plausibility check of the results: quartsclient.pl	[GPL]

Table 6 ST4.2 Client Software

Comment: This requirement is fulfilled by the QUARTS GUI.

req-st4.2-4: The software in Table 7 is currently required to be installed on the server.

Intensity inhomogeneity correction software: vsegment3dpar	[GPL]
Linear registration software: vreg3dpar	[proprietary MPI]
Intensity adjustment software: vintens3d	[proprietary MPI]
Non-linear registration software: miafluid3dpar	[GPL]
Applying deformation field: miaassignshift3d	[GPL]
fMRI to aMRI data overlay software: vpredisp	[proprietary MPI]

Table 7 ST4.2 Server side software

Comment: Names of the software components changed compared to D1.1. Additional software vsegment3dpar, vintens3d and vpredisp is required on server side.

In comparision with D1.3a, the following progress has been made:

QoS: A performance prediction model has been implemented for the preprocessing and processing stages of QUARTS including the prediction of runtime, main memory and disk space. It's based on average tables of recorded values obtained from multiple test runs with sample datasets. The necessary parameters are extracted from the input data on the client side and converted into a GEMSS request descriptor.

Ultrasound: The existing tool chain has been examined to its possible extension to operate with ultrasound (US) data. The analysis of algorithms for MR to US registration, which forms an essential step in the image processing chain, showed that currently there is no reliable, robust and automatic method known and further research would heavily exceed the frame of the project.

GUI: A graphical user interface has been added to the client. It allows the selection of the necessary input data and the computation backend. The 3 different computation targets allow execution of the chain on the local machine, a local PC-cluster or a GEMSS phase 2 grid site. During chain execution the user will be informed about the current status. After the computation has finished the result data will be automatically transfered to the client and visualised. The visualisation can be interactively customized by the user.

Due to a shift of priorities, the GEMSS middleware was unable to assure the immediate start of a server side enqueued software component at the original release time of this deliverable. This issue has been addressed, and it is now possible to run the ST4.2 service under real-world conditions in a useful manner.

3.2.3 ST 4.3 RAPT - Stereotactic Radiosurgery Simulation

Intracranial lesions are routinely treated using Stereotactic Radiosurgery, a technique that uses intense levels of gamma radiation to control a tumour or vascular abnormality within the brain. The use of many sources firing convergent beams from many directions gives a high radiation dose within the lesion whilst minimising the dose to the surrounding normal brain.

Monte Carlo approach

Determination of the radiation field necessary to treat a lesion in the brain is performed by numerical solution of radiation transport equations designed to deliver the target dose specified by the clinical consultant. The use of an efficient parallel Monte-Carlo code running on the Grid is intended to enable high accuracy treatments to be planned and executed within specified time

constraints. It is not practical to use Monte-Carlo for every simulation, since compute times are too long (even with the Grid). Experience within GEMSS has demonstrated that the Grid is best used in a supporting role, verifying clinical treatment plans by using the sophisticated Monte Carlo physics model provided by the ST4.3 application (RAPT). Clearly it is impossible for an individual clinical department to justify installation of a dedicated supercomputer for treatment planning support of this nature. Only by outsourcing can a Monte-Carlo approach be used on demand, and only by doing so in a managed Grid environment can it be delivered with sufficient timeliness and quality of service.

Fulfilment of Original Requirements Specification

The following application requirements are reproduced from the Requirements Specification deliverable D1.1.

req-st4.3-1: A workstation is required to run the client application software.

req-st4.3-2e: The NHS patient record must be updated when the results of the simulations are returned. (E)

Comment: This is a long term objective, beyond the scope of this project

req-st4.3-3: Visualization software capable of dynamic iso-surface extraction is required.

req-st4.3-4: A mechanism to upload patient target position, MRI scan data in DICOM format. Head geometry contours are required. These data transfers will be up to 1Mbyte, 100Mbytes and 5 Mbytes in size respectively.

Comment: In the clinical setting, head geometry is not obtained from MRI, but derived from 24 measurements that characterise a parametric head. This 'bubble head' is the 'gold standard' for defining skull geometry in Gamma Knife radiosurgery and is fully accommodated within the RAPT application. The simplicity of the head geometry format in combination with zip file compression has reduced the size of data transfers to less than 10MB.

The quality of service requirements are as follows:

Service completion time	Up to 10 minutes
Service Throughput	1-2 per day
Service Security	Anonymisation, encryption, authorisation, authentication
Service Availability	High: >99%

Comment: For iterative treatment planning, turnaround times of less then a minute would be required. Given the scalability limit of RAPT at roughly 50 processors and the limitations of today's processing power, this can currently not be achieved. A more appropriate role is to validate the solutions obtained with GammaPlan, and to recommend alternative treatment strategies in those cases in which GammaPlan is found wanting (eg. when skull material homogeneity is evident). In this way Grid-enabled RAPT can have significant impact on clinical patient management.

A revised table of requirements is proposed as follows

Service completion time	Less than 60 mins (but the shorter the better!)
Service Throughput	1-3 per day (increase due to clinical desirability of service)
Service Security	Anonymisation, encryption, authorisation, authentication
Service Availability	High: >99%

req-st4.3-5: The software is currently required on the client workstation. Interfaces between these and the RAPT code will need to be developed in GEMSS:

Radiosurgery planning software GammaPlan	[proprietary Elekta Instruments]
HP graphics and visualization software	[proprietary HP]

Table 8 ST4.3 Client side software

Comment: A GUI for the RAPT client has been completed, and permits replication of clinical radiosurgery plans and their visualisation in RAPT. This requires MatLab on the client.

Monte Carlo code, based on IT Innovation RAPT project code	[proprietary IT Innovation]

Table 9 ST4.3 Server side software

req-st4.3-6: The software is currently required to be installed on the server.

3.2.4 ST 4.4 COPHIT - Inhaled Drug Delivery Simulation

The inhaled Drug Delivery Simulation utilises a respiratory model that integrates medical images, mesh generation, Computational Fluid Dynamics (CFD), compartment modelling of the lungs, and the simulation of inhalation devices. Overall, it is a comprehensive simulation tool for the study of new lung treatments and drug delivery to the lungs, with a focus on determining the conditions necessary for targeted delivery of medication.

Original GEMSS Application Profile

The originally envisaged use scenario for the ST4.4 software was that it was part of a service that would begin with CT data (imported from a hospital), segmented and used to generate a lung mesh to define the geometry of the airways. Boundary conditions for the model would be specified (inhaler device performance and drug characteristics) and the simulation run to obtain a timely depiction of flows and drug deposition within the airways, including subsequent clearance through the systemic circulation.

Comment: This generic profile offers the user great flexibility, but its breadth results in an expansive application that is daunting to the user. In an effort to make the simulation more accessible, the EASA GUI is used to hide much of the complexity of the application, but it comes at the cost of some loss of flexibility. Nonetheless these changes have met with the approval of the end users. Consequently, the application profile has changed and now begins with the end-user (eg. pharmaceutical manufacturer or consultancy) specifying the physical and pharamacokinetic properties of an inhaled drug, together with an inhalation regime (flow profile, etc.). The simulation is run to obtain a timely depiction of flows and drug deposition within the airways, including subsequent clearance through the systemic circulation, for a number of different airway geometries (e.g. adult, child, diseased) which are stored on the Grid server. The COPHIT application does not require shell access by the user (see **req-s10** of D1.1). The Grid implementation of this software manages billing in accordance with the degree of Grid utilisation and safeguards data integrity and security.

Fulfilment of Original Requirements Specification

The following application requirements are reproduced from the Requirements Specification deliverable D1.1.

req-st4.4-1: A workstation is required to run the client application software.

req-st4.4-2: The COPHIT code must be partitioned into client and server parts prior to Grid operation.

req-st4.4-3e: The NHS patient record must be updated when the results of the simulations are returned. (E)

Comment: Long term goal. Initially Cophit will be targetted at device manufacturers / pharmaceutical companies to optimise device design / drug formulation and therefore integration with Hospital Information Systems is not a priority.

req-st4.4-4: A rich graphical interface is required to display complex 3D geometries. This could be provided by CFX/EASA software.

Comment: The use of CFX and EASA has been adopted for this purpose

req-st4.4-5: A mechanism to upload CT scan, inhaler specification, drug specification, target area specification and a patient image database is required. Data transfers of up to 300Mbytes will be required to the server and 1.5GBytes from the server.

Comment: The revised application profile resulting from the use of EASA means that the end-user is able to access a database of airway geometries stored on the Grid server. There will not be a mechanism for users to generate their own CFX meshes from CT scans, rather these will be uploaded by the application developer. (This process involves a wide range of possible data types - e.g. CT, MRI, CAD file - typically beyond the resources and expertise of the end-user, but this service could be provided by a consultancy company). The presence of a library of files on the Grid server significantly reduces the input data transfer size to a few kilobytes. The results report generated using EASA is also much smaller in size than the full set of CFX results (which could exceed 1Gbyte). However, sufficient flexibility remains that some or all of the raw CFX results could be downloaded if the user desires.

The quality of service requirements are as follows:

Service completion time	1-2 weeks
Service Throughput	1-2 per day
Service Security	Encryption, authorisation, authentication
Service Availability	90%

Table 10 ST4.4 QoS requirements

Comments: The normal maximum job length on the NEC cluster is 4 hours, but this can be increased for problems which require long solution times. One second of simulated time requires many hours of compute time (parallel, 8 processors), which indicates that inhalation/exhalation studies can be performed in a few days.

req-st4.4-6: The software in Table 11 is required on the client workstation.

MATLAB 5.3	[commercial software – The Mathworks]
Ensight (or CFXpost) engineering analysis post-processor and display software (client-server version)	[commercial software - CEI] [CFXpost – AEA Technology]
Text editor (Excel spreadsheet editor also useful)	[commercial software – eg. Microsoft]
CFX (CFXbuild and associated tools)	[proprietary - AEA Technology]
GUI – to oversee data transfer (needs to be created)	[proprietary – GEMSS]

Table 11 ST4.4 Client software requirement list

Comments: The end user relies on EASA client software to establish connection with the Grid through an EASA server Grid portal. Therefore an EASA Client (AEA technology) is required. The software to oversee data transfer is installed on a GEMSS client portal computer, and the EASA Server. The presence of EASA removes the need for MatLab on the client, and provides a non-interactive results visualisation alternative to Ensight or CFX-Post.

req-st4.4-7: The software in Table 12 is required to be installed on the server.

MATLAB 5.3 / Simulink 3.0.1	[commercial software – The Mathworks]
Ensight (or CFXpost) engineering analysis post-processor and display software (client-server version)	[commercial software - CEI] [CFXpost – AEA Technology]
COPHIT v1.2.2	[proprietary – COPHIT]
CFX 5.5.1 computational fluid dynamics solver	[proprietary AEA Technology]
Segometex - segmentation software	[proprietary Mainz]
Fortran/C compilers	[commercial software]

Table 12 ST4.4 Server software requirement list

Comments: MATLAB/Simulink for the Grid server is no longer required (replaced by FORTRAN routines). CFX 5.6 is now being used. EASA enabled Cophit replaces Cophit v1.2.2. The presence of mesh libraries on the Grid negates the need for Segometex (a tool for CT scan import, relevant to mesh generation).

Current Status Summary:

The presence of EASA has changed the profile of the application profile dramatically and therefore several requirements have undergone significant changes, parts of which are now redundant. The advantage of EASA is seen to be the creation of a much more streamlined and focussed application, and the consequence of providing mesh libraries on the Grid means that data transfer can be much reduced. Furthermore, mesh generation is removed from the client and becomes the responsibility of the portal provider who is much better placed to provide such a service. QoS and business models are included.

3.2.5 ST 4.5 CARDIO – A Simulation of the Cardiovascular System

Simulation of the cardiovascular system is already a valuable tool in the development of prostheses. Analysis of the haemodynamics of cardiovascular devices such as prosthetic heart valves is now routinely used as part of the development process. Structural finite element analyses support the design process, particularly for devices such as tissue valves. Most recently, coupled fluid-solid interaction simulations have been developed, recognising the complex interplay of the fluid and structural **elements** of the system. A potentially fruitful application of this development will be the facility to answer ‘what if’ questions in the context of an individual patient. Its role in surgical planning will develop as run times reduce and studies validated against in vivo data are presented in the clinical literature. Generally cardiovascular simulations currently operate either at a global level, typically in terms of a one-dimensional electrical network analogy, or at a local level by detailed three-dimensional computational fluid dynamics with prescribed boundary conditions. In GEMSS these levels of model are coupled to provide global simulation capacity with the facility for local detailed analysis.

GEMSS Application

The cardiovascular system simulator has much in common with the COPHIT application. It features both three-dimensional and one-dimensional compartments. In parallel with the COPHIT application, the Grid must address user-interface, accessibility, bandwidth, processor scalability and storage issues to provide an efficient solver for the computational fluids component of the simulation.

Fulfilment of Original Requirements Specification

The following application requirements are reproduced from the Requirements Specification deliverable D1.1.

req-st4.5-1: A workstation is required to run the client application software.

req-st4.5-2: The cardiovascular code must be partitioned into client and server parts prior to Grid operation.

req-st4.5-3: A database of patient images will need to be either transferred across the Grid to run the job or pre-installed at the Grid server. The legal issues involved with this must be addressed before real patient images can be released for use in the test-bed.

Comment: No intention to store patient images on the Grid server. CFX meshes are stored instead.

req-st4.5-4: A rich graphical interface is required to display complex 3D geometries. This could be provided by CFX/EASA software.

Comment: CFX is used for 3D flow visualisation. Ease of use issues are addressed by using a MatLab GUI rather than EASA.

req-st4.5-5: A mechanism to upload device specification, target area specification and a patient image database is required. These data transfers will be up to 300Mbytes are required to the server and 1.5Gbytes from the server.

Comments: The end-user has access to a database of mesh geometries stored on the Grid server. If the end-user has the facility, then they can provide their own geometries as well, but this should be in the form of a CFX mesh, rather than the original CT or MR scan data. Input mesh data is typically <20MB in size, and the input data size is further reduced if a server-stored library mesh is used.

The quality of service requirements are as follows:

Service completion time	1-2 weeks
Service Throughput	1-2 per day
Service Security	Encryption, authorisation, authentication
Service Availability	90%

Table 13 ST4.5 QoS requirements

Comments: The normal maximum job length on the NEC cluster is 4 hours, but this can be increased for problems which require long solution times. One second of simulated time requires many hours of compute time (parallel, 8 processors), which indicates that a study over several cardiac cycles can be performed in a few days.

req-st4.5-6: The software in Table 14 is currently required on the client workstation.

MATLAB 5.3	[commercial software – The Mathworks]
Ensight (or CFXpost) engineering analysis post-processor and display software (client-server version)	[commercial software - CEI] [CFXpost – AEA Technology]

Text editor (Excel spreadsheet editor also useful)	[commercial software – eg. Microsoft]
CFX (CFXbuild and associated tools)	[proprietary - AEA Technology]
GUI – to oversee data transfer (needs to be created)	[proprietary – GEMSS]
ANSYS structural analysis software	[commercial software – ANSYS Europe]

Table 14 ST4.5 Client software requirement list

Comment: Software has been tested using MATLAB 5.2 and MATLAB 6.5. CFX-Pre is required for generating new meshes and CFX-Post is required for results visualisation.

req-st4.5-7: The software in the Table 15 is currently required to be installed on the server.

MATLAB 5.3 / Simulink 3.0.1	[commercial software – The Mathworks]
Ensight (or CFXpost) engineering analysis post-processor and display software (client-server version)	[commercial software - CEI] [CFXpost – AEA Technology]
COPHIT v1.2.2	[proprietary – COPHIT]
CFX 5.5.1 computational fluid dynamics solver	[proprietary AEA Technology]
SimBio software toolkit	[proprietary - SimBio]
Fortran/C compilers	[commercial software]

Table 15 ST4.5 Server software requirement list

Comments: CFX 5.6 now in use and only CFX-Solve is required for the CFD solution step. ANSYS can reside on the Grid server (but is only required for coupled fluid-solid interaction simulations). The coupled compartment sections of the COPHIT software have been completely rewritten from scratch for the cardiovascular application. COPHIT is not used in its original form (however, certain methods used in the original code have been copied and adapted). MATLAB is not required on the grid server (replaced by FORTRAN routines). The SimBio tools have been used to aid the production of some meshes, but they are used on the client computer rather than the server because of the high level of interactivity required.

3.2.6 ST 4.6 SPECT - Advanced Image Reconstruction Service

Visualisation of the distribution of radio-pharmaceuticals by Single Photon Emission Computed Tomography provides valuable complementary information to the representation of anatomy from high-resolution imaging modalities such as x-ray CT and magnetic resonance imaging. Modern fully 3D iterative reconstruction algorithms provide enhanced image reconstruction for the whole image volume, by considering principal 3D effects of data acquisition. The image reconstruction service offers improved reconstruction of 3D SPECT data as demonstrated by the Grid test case. Evaluation issues include assessment of its operation in a clinical environment and the implementation of the GEMSS security infrastructure.

Within the GEMSS project, fully 3D image reconstruction was grid-enabled from a stand-alone parallel application into a flexible Grid service, providing an easy to use, intuitive client front-end together with a fully transparent Grid-service part. The java-based graphical user interface (GUI) was implemented as an ImageJ plugin providing a vast variety of image processing functionality.

Since data compatibility with imaging modalities of different vendors is a strong criterion for the wide-spread usage of the reconstruction services it is important to provide an easy-to-use tool to

import/export DICOM image and projection data. As a part of the DICOM standard the import/export of image and projection data has been implemented with the GUI.

As part of the GEMSS QoS support an accurate application performance model based on measured reference times was implemented. It provides robust estimations of prospective server side execution times and acquired system resources which are a basis for flexible business models.

Fulfilment of Original Requirements Specification

The following application requirements are reproduced from the Requirements Specification deliverable D1.1. A tick in the margin indicates compliance with the originally drafted specification whereas a cross indicates non-compliance. In the case of the latter, a comment is typically supplied to clarify the status of this item.

req-st4.6-1: A workstation is required to run the client application software.

Comment: Client installed on work station and notebook.

req-st4.6-2: A mechanism to upload DICOM medical image data to the client workstation is required. These data transfers will be up to 4Mbytes in size.

Comment: GUI provides DICOM import/export tool.

The Quality of Service requirements are as follows:

Service completion time	Depending on scenario (see IBMTP): few minutes, few hours, few days
Service Throughput	50 requests/day
Service Security	anonymization & encryption
Service Availability	90 %

Table 16 ST4.6 Quality of Service Requirements

Comment: The ongoing evaluation shows that these requirements are met by the GEMSS middleware and the ST4.6 client software.

req-st4.6-3: The software in Table 17 is currently required on the client workstation.

ImageJ	[open source software]
Reconstruction client	[proprietory IBMTP]

Table 17 ST4.6 Client software requirement list

Comment: Matlab and Analyze are not needed anymore, ImageJ provides image processing functionality. The end-user benefits from the fact that there are no licence fees to pay any longer. The GUI can be used as a stand-alone applications or as an ImageJ plug-in.

req-st4.6-4: The software in Table 18 is required to be installed on the server.

Reconstruction server (Java & ANSI C, MPI, OpenMP)	[proprietory IBMTP]

Table 18 ST4.6 Server software requirement list

3.3 Middleware run-time testing

This section reports on a number of middleware evaluation activities that have been conducted by the GEMSS project. Each evaluation activity focussed on a different aspect of the GEMSS

infrastructure, with the aim of providing a complete picture of how the GEMSS infrastructure performs.

After the completion of each development phase the middleware is released and a stress test conducted with the aim of testing to destruction. This allowed the middleware developers to quantify the robustness of each phase of the infrastructure, and helped with the discovery of errors that could subsequently be fixed. It also shows how the system performance degrades under failure. We report the results from three stress tests, conducted for phase I2 (February), phase I3 (October, November) and phase I4 (November).

Several of the features of GEMSS come with a performance penalty. For the final phase I4 release we have evaluated the performance overheads associated with the security features of GEMSS and the negotiation process employed in GEMSS.

A key aspect of the GEMSS design is the use of application specific performance models. These performance models reside on the server side and allow the client code to upload a small metadata description of a job prior to negotiation. The performance model code estimates the actual job execution time based on this small metadata description. Job execution estimates are used to make reservations on the service provider's cluster, which allow a number of CPU's to be reserved for each job. We evaluate the performance model error, and assess the safety margins built in to each performance model to ensure a big enough scheduler reservation is made.

One of the most important requirements for the GEMSS infrastructure is robustness. The robustness of the final phase I4 infrastructure is measured by computing the error rates that occurred when using the infrastructure over a specified period of time.

This section concludes with comparisons to other Grid projects in the literature.

3.3.1 Performance metrics

The following metrics have been used in this whole section:

Error rate

The error rate is a useful measure to reveal the number of jobs that were not successful as a proportion of the total number of jobs. Since robustness is a key requirement of the GEMSS infrastructure we are looking for a very low infrastructure error rate to indicate a robust system.

$$Error\ rate = \frac{(total\ jobs - successful\ jobs)}{total\ jobs}[no\ unit]$$

Performance Model

The performance model error is measured as a proportion of the total execution time. A good performance model will always have a safety margin built in so that the estimated time is greater than the actual execution time. This safety margin makes sure that the scheduler reservation made for a job is big enough, and that the job will not be terminated prematurely.

$$Performance\ model\ error = \frac{(estimated\ exec\ time - actual\ exec\ time)}{actual\ exec\ time}[no\ unit]$$

Negotiation Overhead

The negotiation overhead is measured as the time taken from the very start of the negotiation process, where the service provider is contacted for the first time, to the very end where the user accepts the contracts and the client code informs the winning service provider.

$$Negotiation\ overhead = (negotiation\ end\ time - negotiation\ start\ time)[seconds]$$

Data Transfer Rate

Data transfer rates quantify the speed of data download (and of course upload) given the transport protocol and security protocols employed.

$$Data\ transfer\ time = time\ taken\ [seconds]$$

$$Data\ transfer\ rate = \frac{number\ of\ bytes\ transferred}{Data\ transfer\ time}\left[\frac{bytes}{seconds}\right]$$

3.3.2 Stress Tests

As part of the incremental development strategy employed within the GEMSS project a number of stress test evaluations of each phased release of the GEMSS middleware were conducted. Phase I2 was stress tested in February, phase I3 in October and November and phase I4 in November. These tests attempted to heavily load the middleware and break it under stress; this method can find errors that only occur under a high load or that are intermittent in nature (memory leaks, locking problems, network errors, etc). The results of each stress test fed directly into the development effort and fixes for the errors found.

Phase I2 (February stress test)

The phase I2 stress test was written up in deliverable D1.3a. The important results are reproduced below:

Data transfer

A cross-European data transfer test was conducted using the phase I2 infrastructure. A number of different sized data sets were uploaded, 10 iterations each, and the average round trip time computed. The round trip time covers upload time, 5 seconds of test service processing, and download time. Table 19 shows the results.

Location	*1k*	*10k*	*100k*	*1m*	*10m*	*100m*	*Network*	*Computer (client)*
Southampton (UK) to St. Augustin (Germany)	21.9s	23.1s	31.1s	37.1s	119.4s (1m 59.4s)	3602.7s (1h 2.7s)	T1 connection, 1.5 Mbit	PC, Win2000, 524M RAM, AMD Athlon (tm) XP 2200+ processor
St. Augustin (Germany) to St. Augustin (Germany)	18.6s	19.4s	18.2s	20.0s	33.9s	129.5s (2m 9.5s)	Ethernet 100Mbit	PC, Red Hat Linux, 2 GB RAM, 2.80GHz processor
Vienna(Austria) to St. Augustin (Germany)	25.0s	19.0s	22.0s	28.0s	80.0s	562.0s (9m 22.0s)	Ethernet 100 Mbit	PC, Win2000, 512MB RAM, 2.26Ghz processor
Vienna (Austria) to St. Augustin (Germany)	25.0s	25.0s	25.0s	27.0s	47.0s	259.0s (4m 19.0s)	Ethernet 100 Mbit	Sun Fire V880, Solaris9, 8GB RAM

Table 19 Data transfer times (round trip) for phase I2

As expected, both the hardware and data size affect the file transfer rates when using the GEMSS infrastructure. At data sizes below a megabyte the processing delays in setting up a connection and performing the appropriate security processing tend to dominate the round trip times. Above a megabyte the network data transfer delays tend to be the dominant factor.

Robustness

Application	*Number of runs*	*Number of network retries*	*Number of job failures*	*Error rate*
FEBiNA (st4.1) client and server at NEC	22	0	0	0
FEBiNA (st4.1) client at Uni. Leipzig, server at NEC	70	1	0	0
QUARTS (st4.2)	62	6	0	0
RAPT (st4.3)	53	171	1	0.02
COPHIT (st4.4)	42	18	0	0
CARDIO (st4.5)	40	10	0	0
SPECT (st4.6)	109	2	0	0

Table 20 Phase I2 robustness test results

A 19 hour robustness test was conducted to test both the applications and infrastructure. The cluster at NEC was cleared of its usual jobs between the period 3.00pm to 9.00am the next day, allowing GEMSS testers full access. All six applications were launched, running in non-interactive batch mode, each running with a single test dataset. Each of the applications submitted jobs sequentially, waiting for the first job to finish before submitting the next, over the 19 hour period. Application services used 16 CPU's per job, except for the local FEBiNA service that used 32 CPU's. With all six applications running concurrently from different European locations, this meant the NEC cluster had to, on average, run four jobs and queue 2 jobs at any one time.

Each application used a test case, chosen to last between 10 and 30 minutes, thus ensuring a reasonable number of iterations were run overnight.

The number of iterations and results status were recorded for each application over the 19 hour period. Table 20 shows these figures. As can be seen there was only a single serious error found. All other errors were network errors, and were recognized by the infrastructure and the network connection re-established automatically and the failed operation retried. This meant that only a single job failed to process. These results are an strong indication of the GEMSS infrastructure's robustness in running secure medical services, something identified as a primary requirement during the requirements capture phase of the project.

3.3.3 Phase I3 (October / November stress test)

During the last week of October a 24 hour stress test was conducted over a Monday and Tuesday. The phase I3 server installation at NEC Germany was used as the service provider, and all six of the applications were run at various partner sites. The logistical makeup of the trial is shown in Table 21.

Role	Location	Technical specs
Service provider	Sankt Augustin (Germany)	Dragon cluster: Intel(R) XEON(TM) CPU 2.20GHz -> dual CPU 8GByte 32x2 nodes AMD Athlon(tm) MP 2800+, 4Gbyte Gigabit ethernet from dragon to all switches All nodes equipped with Fast ethernet (100MBit) and Myrinet 2K (2GBit) Operating system on both nodes and front-end is Fedora Core 2
Client (RAPT)	Sheffield (UK)	PC, Windows 2.4 GHz Intel, 512Mbytes 100Mbps via 3Com 3C920 Integrated Fast Ethernet Controller (3C905C-TX Compatible).
Client (RAPT)	Southampton (UK)	PC, Win2000 524M memory AMD Athlon(tm) XP 2200+ processor T1 connection, 1.5 Mbit
Client (SPECT)	Vienna (Austria)	Sun Fire V880, Solaris9 8GB Ram Fast Ethernet, 100Mbit
Client (FEBiNA)	Sankt Augistin (Germany)	PC, Red Hat Linux GB RAM, 2.80GHz processor Fast Ethernet, 100Mbit
Client (COPHIT)	Sheffield (UK)	PC, Windows 2.4 GHz Intel, 512Mbytes 100Mbps via 3Com 3C920 Integrated Fast Ethernet Controller (3C905C-TX Compatible).
Client (CARDIO)	Sheffield (UK)	PC, Windows 2.4 GHz Intel, 512Mbytes 100Mbps via 3Com 3C920 Integrated Fast Ethernet Controller (3C905C-TX Compatible).
Client (QUARTS)	Leipzig (Germany)	PC, Debian GNU/Linux (Sarge) Kernel 2.6 AMD Athlon MP 1900+ 1.53 GHz 1 GB DDR RAM 100 MBit Ethernet LAN, 10 MBit internet connection

Table 21 Logistical information for the Phase I3 & I4 stress tests

Robustness

Each client application selected a number of jobs to run that would take up to 2 hours on 8 of the server's CPU's. This ensured that the cluster would be stressed, but single application would consume all of the clusters resources. At noon on Monday the application developers ran their client applications, and started to sequentially request jobs to be run on the cluster using the phase I3 infrastructure. All six applications concurrently requested jobs from the server over the next 24 hours, ensuring that the quality of service and scheduler code would be fully stressed.

Two real performance models were used, for RAPT and SPECT, with the other applications using a simple dummy performance model that always returned a fixed execution estimate. All client's used the phase I3 negotiation mechanism, except for the COPHIT client that just used simple job submission without negotiation.

The client application code was instrumented so that statistical information was logged after every run. The server logs were also kept, both to help trace any errors that might occur and to record performance model estimates and actual run times.

During the 24 hour stress test a number of problems with the reservation capability of the phase I3 infrastructure were identified. It was not possible to resolve these reservation problems in time to save the trial, which was stopped after 6 hours. Some applications were left running overnight to help the diagnosis.

After investigation, post 24 hour stress test, it was clear that the problem lay with the reservation features of the COSY scheduler. This reservation problem only appeared under large stress, and hence was not detected before the 24 hour stress test was applied. Reservations could not be correctly confirmed, and hence client applications stopped at the reservation step before the job was uploaded and run. After the October evaluation, a newly integrated scheduling policy was created. This policy was deployed on request of several users who preferred to have their reservations as near as possible to their specified start time. Some experiments showed that after relaxing this criterion moderately the amount of successful confirmations could be strongly increased, even though errors are still occurring. Additionally, synchronisation mechanisms were integrated into the JCOSY Java interface in order to make sure that simultaneous invocations of scheduler functionality can not precipitate the problem. This scheduler error later re-occurred, and a complete resolution of this problem was performed mid-way through the November evaluation.

The reservation confirmation problem occurred on most of the jobs attempted, with only a small fraction successfully confirming a reservation and hence able to move to the job handling phase. A small number of server errors were experienced, being related to the CFX server side component (used by COPHIT and CARDIO) and an intermittent MPI kernel error on the Dragon cluster (used by RAPT).

The robustness of the job-handling infrastructure was equal to that of phase I2, even with the addition of the advanced end to end security code.

Phase I4 (November / December stress test)

During November a week long stress test was conducted, in which all six applications were continually and concurrently run to simulate real Grid usage. The test plan used for this week long evaluation is shown in Table 22. The phase I3 infrastructure was used from Monday to Wednesday, and from Thursday to Sunday the phase I4 infrastructure was employed. This permitted the testing of both versions and their comparison.

Infrastructure	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Phase I3	X	X	X				
Phase I4				X	X	X	X

Table 22 November test plan activities

The November setup was the same as that for October, with the exception that all the applications now had performance models and the phase I3 software had some upgrades to try to overcome some of the problems found in the October stress test. The phase I4 software was used in the last two days of the trial and to schedule jobs to run over the weekend.

Robustness

The experimental procedure for this stress test was for users to run their applications manually during the day, running as many jobs in their test sets as they could. Users would then select a single job and run it in batch mode overnight. This meant that many more jobs would be run, concurrently, overnight and that more normal usage would happen during the day. In this way both aspects of real use, and very heavy use, could be tested.

At the start of the week the same reservation confirmation problem occurred in the October evaluation was evident. However, after some analysis the cause was found to be a MySQL timing error (MYSQL internal locking mechanism failed), and this was fixed on Wednesday in time for further runs. On Thursday the disk space ran out on the cluster, resulting in all overnight jobs failing until Friday morning, when the disk space was re-organised. At the end of Friday the users had stacked up many jobs for the weekend and as the schedule became increasingly full the time to get a reservation, and hence the overall negotiation process, became slower and slower. On analysis this was due to the scheduler algorithm, and a workaround found. The error rates during the week are shown in Table 23, and full details of all software errors found during stress testing can be found in the next section.

Application	Errors			
	MySQL	Server	Infrastructure	Total
FEBiNA (4.1)	69 [88%]	8 [10%]	2 [2%]	78
QUARTS (4.2)	93 [100%]			93
RAPT (4.3)	116 [99%]		1 [1%]	117
COPHIT (4.4)	1010 [99%]	1 [1%]		1011
CARDIO (4.5)	575 [99%]	5 [1%]		580
SPECT (4.6)	310 [98%]	6 [2%]		316

Table 23 Error analysis for entire week

If the reservation confirmation errors due to the (solved) MySQL locking failure are excluded from the analysis, it is evident that the basic job handling infrastructure for running a secure job over the Grid performs well. Once the client software had actually managed to get a reservation confirmed the success rate was actually very high. Table 24 show the error rates for all the weeks' jobs, and for those jobs where a reservation was successfully confirmed and hence able to proceed to the job-handling phase.

Application	Total jobs run	Av Error Rate		
		All jobs	Exclude MySQL locking errors	Exclude MySQL locking & server errors
FEBINA (4.1)	143	0.545	0.121	0.029
QUARTS (4.2)	220	0.418	0	0
RAPT (4.3)	147	0.795	0.117	0.032
COPHIT (4.4)	1057	0.956	0.041	0
CARDIO (4.5)	690	0.840	0.043	0
SPECT (4.6)	417	0.757	0.098	0

Table 24 Error rates for entire week

This result verifies what was observed during the phase I2 stress test demonstrating robust behaviour for basic job handling, and indicates that the new end to end advanced security software is robust too. The errors found following confirmation of a reservation were almost all server errors, not from the applications or the infrastructure but from third party software installed on the server (such as MPI). This underlines the fact that basic system administration is a key issue when running

a cluster, and no matter how good the Grid infrastructure is the system administrator will always play a key role in maintaining robustness.

Software improvements as a result of stress testing

Phase I2

The single job that failed in February was due to a network connection failure while parsing a SOAP message (the parser failed to load a remote resource required for parsing the XML SOAP message). This was actually an error within a third party code library, and as such was not trapped by our infrastructure. This fault was, however, identified and future versions of the infrastructure trapped this network error.

Phase I3 & I4

The majority of all errors were caused by the reservation confirmation problem due to the MySQL locking failure (MySQL is project external software!) which disappeared after a workaround has been implemented in the GEMSS software.

All reservations were properly processed and integrated into the queuing logic, but it was often not possible to confirm a valid reservation within the requested period of time. The implementation of the confirmation procedure simply changed an entry within a MySQL database table, but due to several race conditions (cf. D3.2b) of the involved threads, the "new" value of this element wasn't available on time. The source of this error was finally discovered on Wednesday during the November stress test, and a minor software change was sufficient to overcome the problem.

Due to the installation of a large software package on the cluster shortly before the start of the November evaluation, the root file system ran out of space during the overnight batch tests on Thursday. As a consequence, all programs that were using this partition for I/O operations failed immediately. The scheduler, for example, was no longer able to store intermediate files into the /tmp directory and thus no reservation could be made from that point in time. After having detected this error the file system structure was re-organized in order to have a sufficiently large partition available.

Whilst loading up the scheduler on Friday with many reservations for over the weekend, it was noted that the scheduler reservation algorithm became progressively slower as more and more reservations were made. This is an inherent feature of the scheduling algorithm, and forced the users to eventually give up reserving jobs since it took too long. There is no simple cure for this problem. A workaround was to move the requested execution window further into the future, avoiding large congested areas of the current job schedule.

3.3.4 Performance evaluation

The GEMSS infrastructure supports a number of technical features, such as security and quality of service negotiation. These technical features are essential for the GEMSS Grid since we are dealing with Medical data, and application domains that require computing resources to be available at well defined times. However, both security and negotiation have a performance penalty associated with them, and this penalty will limit their value to certain situations and certain application domains.

Security overheads

There are two basic security mechanisms available to the GEMSS Grid. The transport level security mechanism is HTTPS and secured SOAP attachments, and this is the minimum requirement for secure data transmission over the Internet. Since the data processing by GEMSS includes medical data, it also supports an additional, more advanced security mechanism, involving end-to-end security. Full details of these technical features can be found in the design deliverable D1.2b. Since SOAP attachments are used there is a basic limit of 2Gbytes to all data transfers.

The basic data transfer speeds were measured with and without the advanced end-to-end security for a variety of different data sizes. These tests were conducted from a client site in Southampton to a server site in Germany using the GEMSS Grid, with client and service provider hardware specifications identical to the October stress test setup. Ten iterations were run for each data set, and the average download times measured both with and without end-to-end security. Figure 5 shows the results.

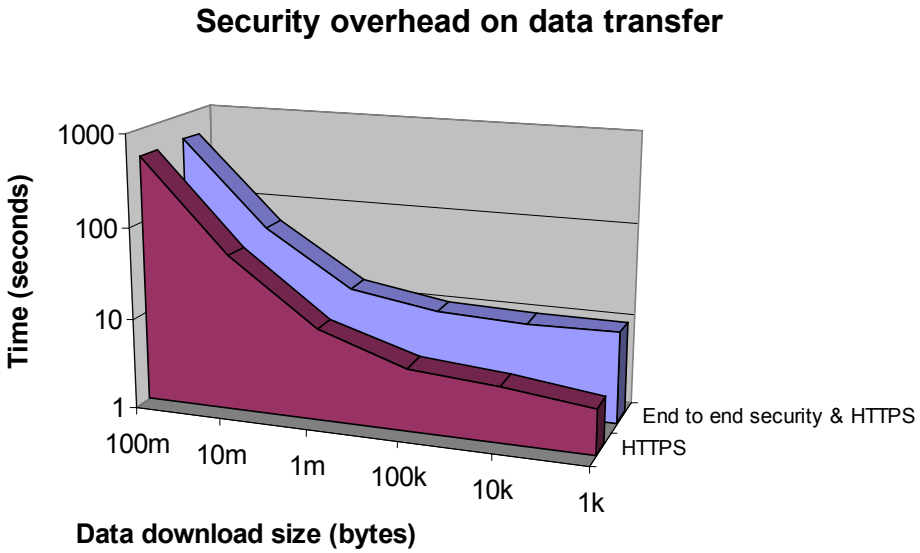

Figure 5 Security overhead on data transfer (phase I4)

As can be seen in Figure 5 there is about a 6 second additional overhead associated with the end-to-end security protocol. This is insignificant when the data size is large, since the basic network overheads dominate the transfer times. However, for jobs where the data size is very small then this overhead will play a major role, and hence should be a major consideration for any application that runs multiple small jobs. It would be much more efficient to package up small jobs and run them as a single aggregate job.

It should be noted also that all GEMSS web service operation invocations suffer from these security overheads, and that subsystems like the negotiation component, which makes many web service operation invocations, accumulate these overheads to the extent that they become a significant aggregate overhead. Basic job handling is largely unaffected since the vast majority of the time is spent uploading and downloading the data files.

Negotiation overheads

There are two levels of negotiation within the GEMSS infrastructure. At the lowest level a single service provider will perform a negotiation with the underlying scheduler to find the best slot that meets the clients requirements. This involved trying a set of node configurations, typically 2,4,8 and 16, and finding out what start, end and price will result.

At a higher level there is an advanced negotiation between the client and several service providers via a client driven reverse English auction protocol. The client will, for a number of auction rounds, ask each service provider to make a bid to run a job; bids are made in the form of a web service level agreement (WSLA). A threshold score is set for each auction round, below which no bids will be considered. After each round this threshold is increased until only a single service provider is able to meet the threshold. This auction has a significant negotiation overhead, but will improve the quality of the WSLA contracts the client receives.

The negotiation time is roughly proportional to the number of bidding rounds, as can be seen in Figure 6. The score of the WSLA, related to the weight and value of each of the quality of service parameters specified by the client, also improves as the rounds go on and the threshold value is linearly increased; this is shown in Figure 7. Since the service providers are competing, the owner of the best scoring bid per round will tend to change round to round. The winner on the last round is selected by the client to run the job.

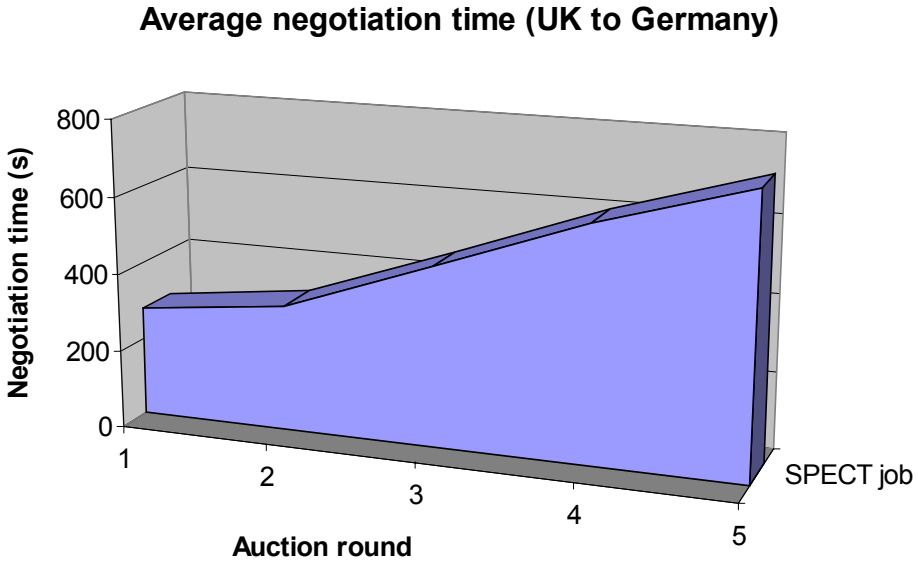

Figure 6 Negotiation time VS number of auction rounds

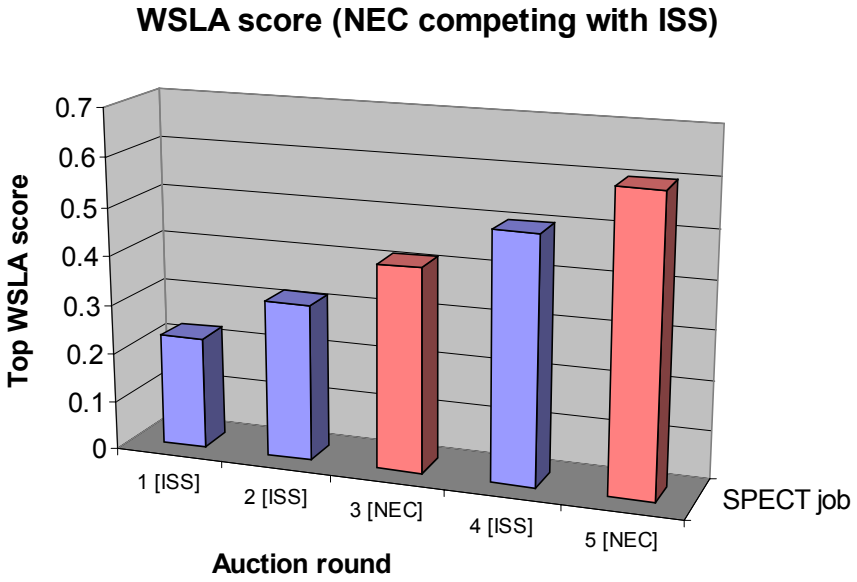

Figure 7 WSLA score during the auction

The negotiation time per auction round is configurable by the client software. The choice of this duration is dependant on the expected network and machine performance of the set of service providers in the auction. For example, if the average time of a service provider to perform its internal scheduler reservation is 2 minutes, the client would want to wait about 3 minutes before ending the auction round. If the auction round ends before the service provider is in a position to make a bid, and hence return a WSLA, then that service provider is removed from the auction and the other services providers will be chosen in preference. Some example negotiation scenarios and actual timings are listed in Table 25, taken from the Southampton to Vienna / Sankt Augistin test runs performed doing the robustness tests.

Example	Auction rounds	Typical negotiation time
2 service providers 1 client	5	10 minutes
1 service provider 1 client	3	7 minutes
2 service providers 2 clients	3	9 minutes

Table 25 Example negotiation overhead times for different setups

Advanced negotiation is deployed as a technology demonstrator in GEMSS and work is underway to evaluate it more fully before the end of the project.

3.3.5 Performance model evaluation

Both the phase I3 and phase I4 infrastructure quality of service code use application specific performance models to estimate the execution time of a job based on a description of that job (job metadata size < 10kBytes). These execution time estimates are then used to reserve a scheduler time slot, via an iterative quality of service negotiation with the scheduler, which can then be agreed and confirmed by the client and made permanent. Only when a reservation has been agreed will the actual input data be uploaded to a service provider and the job run by the scheduler.

Performance model estimates were recorded server side over the whole week, along with actual execution times for each job. The accuracy of each application's performance model was computed, averaged over the data set used over the whole week. Table 26 contains figures for each application's performance model, and this is graphed in Figure 8.

Application	Total jobs run	Number of job types	Average error (estimate - real) / real	Standard deviation	Confidence interval
FEBiNA (4.1)	57	3	1.293	0.327	0.085
QUARTS (4.2)	128	2	0.023	0.148	0.025
RAPT (4.3)	26	19	0.313	0.306	0.123
COPHIT (4.4)	7	2	1.343	0.090	0.083
CARDIO (4.5)	42	2	0.286	0.158	0.047
SPECT (4.6)	69	6	0.037	0.031	0.007

Table 26 Average performance model accuracy figures

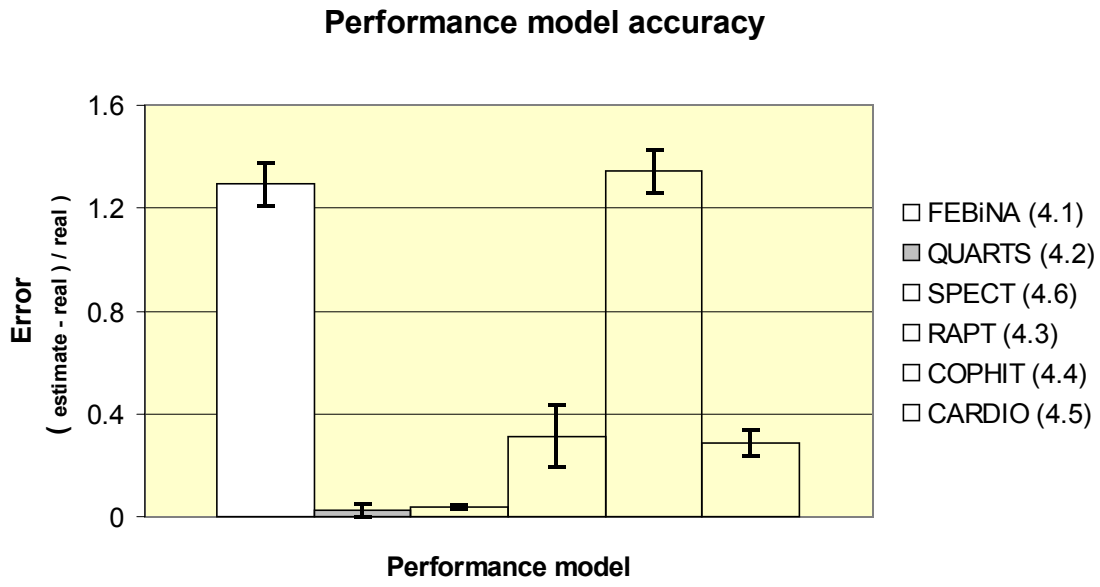

Figure 8 Average performance model accuracy

As can be seen in Figure 8 the quality of the performance model is very dependant on the application type. Some applications, such as RAPT, are linear in nature and hence are relatively easy to predict. Other applications, such as COPHIT, are non-linear and are almost impossible to predict. The key to designing a performance model is to built in enough of a safety margin so that the jobs execution time is never underestimated. If a job execution time is underestimated then the reservation made on the cluster scheduler will be too small, and hence the job will be killed before it

is finished. This is the reason all performance models have an average error greater than zero, and in some cases quite a lot larger.

For the FEBiNA service (ST4.1) the errors in the run time prediction have been closely analysed. It has been shown, that the run time estimates of the Finite Element code showed rather small errors between 0.1 and 0.2. The large errors around 1.2 shown in Figure 8 are due to an update of some I/O routines of the cluster operating system that has not been incoporated in the performance model at testing time and that enormously accelerated the DRAMA mesh partitioner which was used to pre-process the mesh. In the special test case the DRAMA partitioner needed 70 seconds of computing time after the I/O update, while it used 249 seconds at the time when the performance model was created. In general, file I/O can become a source of uncertainty for the runtime prediction, since large simultaneous file I/O can easily become a bottleneck for most HPC platforms and is merely unpredictable for a performance model based on a single application.

For applications that are genuinely hard to predict the basic reservation mechanism / simple prediction model supported by GEMSS is not sufficient. A possible option is to apply a policy allowing a job to terminate early, freeing up the resources, and the charges to reflect this. Another option is an advanced predictions model that allows the prediction of complex simulations based an previous experience (database approach).

3.3.6 Robustness evaluation

In addition to the stress testing that has been performed, where abnormal stress is applied to the server in an attempt to trigger errors, a robustness test was performed to measure more “normal” usage over the period of one day; by “normal” is meant running jobs manually while other non-GEMSS users are allowed to run jobs on the cluster. This test was conducted with multiple clients and multiple service providers. The clients were run from Southampton and Vienna, and the service providers run at Sankt Augustin and Vienna; the hardware configurations are as per the October stress tests.

A small set of manually run SPECT jobs was run for each of the N client N service provider configuration options. The error rates and average negotiation time figures are shown in Table 27.

Number of clients	No. of service providers	Jobs run	Successful jobs	Error rate
1	1	6	6	0
2	1	10	10	0
1	2	8	7	0.125
2	2	10	10	0

Table 27 N client N service provider robustness test results

As can be seen in Table 27 the phase I4 infrastructure is robust under normal conditions. There are rare occasions when making a reservation on the server is difficult due to the non exclusive use of the clusters. In a multi-user environment including non-GEMSS users on the cluster it happened that nodes had been occupied outside the GEMSS Resource Management before the GEMSS CRM was able to confirm the reservation. This is an operational problem that can easily be solved and not a problem of the GEMSS middleware.

3.3.7 Summary

The GEMSS middleware evaluation has concentrated on the robustness and security of the GEMSS infrastructure. These aspects are key requirements for the GEMSS project, and on both accounts the middleware performs well. As a result of three separate stress tests, running over many days of real Grid activity, the robustness of the infrastructure has been improved, culminating in a final phase I4 evaluation that demonstrated low error rates.

The end-to-end security and transport level protocols have proved to be very robust in operation. The performance penalties associated with these two security protocols have been quantified, and it is clear that it is much better to run a single large aggregated job than many small jobs with small data sizes.

The GEMSS negotiation overheads are of the order of minutes, so for real usage there will be a trade off between the cost savings from improving the score of the WSLA and the time taken to negotiate this better deal. If the application needs to run many small jobs quickly then there is no real benefit in negotiating for more than one round (of perhaps at all), but is the application jobs are large then it is worth investing some time in getting a superior deal from a service provider.

The negotiation infrastructure has proved to be reliable, limited only by the underlying cluster scheduler system. Scheduler reservation has proven difficult but significant progress in investigating this in the context of GEMSS has been made and this experience has resulted in improvements in the COSY scheduler system.

The stress testing demonstrated that good system administration is a key (but mundane!) aspect of running a robust Grid. Simple things like running out of disk space or having unstable network connections will drastically affect the robustness of an otherwise very sound infrastructure. For real use it is clear that well formulated procedures would be needed when running a Grid resource to ensure downtime was at a minimum.

The GEMSS middleware needs application specific performance models for each of the applications it supports. Some application types are well suited to performance modelling, being linear in nature of easily predicable. However, some applications are inherently unpredictable and other mechanisms for performance estimation are needed in these cases, such as over-reserving and releasing resources as the become available when a job finished.

The robustness testing with N client N service providers demonstrates just how robust the GEMSS infrastructure is under normal conditions. However, it is not perfect, and it is at the mercy of the Internet and local cluster users who might suddenly saturate the cluster with large jobs. Nevertheless, on a well organised cluster with a disciplined approach to usage there is every indication that the GEMSS infrastructure would work well.

3.3.8 Security checklist

The security deliverables D2.2a and D2.2c provide a detailed analysis of the state of the art in Grid security. The following security features have been considered within the GEMSS infrastructure, detailed in Table 28.

Defence type	GEMSS design feature	Implementation progress
Internal audits	GEMSS specific additions to partner site's existing internal audit procedures	Exploitation
Independent verification	Third party verification of GEMSS security	Exploitation
Risk management protocols	Formally drafted risk management procedures	Exploitation
Exception handling protocols	Formally drafted exception procedures	Exploitation
Acceptable usage policies	GEMSS user policy documents	Exploitation
Public key infrastructure	GEMSS Public Key Infrastructure	Implemented
- Certificates	X.509 compliance, GEMSS CA	Implemented
- Digital signatures	Public Key Infrastructure	Implemented
Secure web protocols	HTTPS, WS Security, End to End security	Implemented
Modular coding	Plug in component architecture	Implemented
Security standards	X.509, HTTPS, SSL, RSA, WS Security	Implemented
Logging	System and auditable logs	Logging in place (both low level system and business account logs)
Security patches	Apache / Tomcat security advisories inform us of patches	Patches up to date
Demilitarized zone	Set-up for GEMSS servers by service providers	Ready
Intrusion detection	NEC planning to use one	Technology demonstrator
Firewalls	Partner site firewalls	Used already by partners

Table 28 Security Features

Of the 11 planned pre-exploitation GEMSS security features, 10 have been fully implemented and 1 will be introduced as a technology demonstrator. There are 5 more features that are appropriate for exploitation, and for these there are guidelines within the security report deliverables.

3.3.9 Long term testing

The aim of the long term tests was to acquire run-time behaviour of the middleware and the applications under real world conditions. The statistics given in Table 29 where gathered within the time frame from the beginning of November till mid December.

long term statistics		*ST 4.1 FEBINA*	*ST 4.2 QUARTS*	*ST 4.3 RAPT*	*ST 4.4 COPHIT*	*ST 4.5 CARDIO*	*ST 4.6 SPECT*
jobs	*issued*	54	250	54	273	368	389
	successful	50	173	12	183	243	371
	in percent	92	69	22	67	66.0	95
avg. Negotiation time (s)		36.0	63.2	34	80.5	84.2	65.5
avg. upload time (s)		57.0	151.6	6	23.0	8	13.5
avg. upload size (MB)		3.8	9.0	0,23	0.023	0.008	0.367
avg. upload speed (kB/s)		67.0	59.5	38,3	1.0	0.35	27.3
avg. download time (s)		10.4	22.3	9	50.1	122.7	4.8
avg. download size (MB)		5.2	6.5	3,1	28.3	74.1	0.15
a. download speed(kB/s)		500.0	293	344	565	604	31,6

Table 29 Long term application statistics

During this period application development was still under way and the server side was still in an experimental stage. Therefore, it is only natural that the percentage of successful runs is low compared to the targeted requirement of a 99% service availability. The majority job failures are server side system errors like "no disk space", MPI socket errors, and problems with the scheduler and not actual application errors.

The average times for negotiation, up- and download describe how the Grid performs under normal conditions (compared to the stress test numbers). These times very much depend on the hardware in use and Internet speed. The difference between up- and download speed is due to an implementation choice of file copy mechanism on the NEC server; this choice is currently under review. Since negotiation times and negotiation times are in the range of minutes, the GEMSS infrastructure is best suited for jobs that require a large run-time, since here these times can be neglected compared to the full run-time of the jobs.

3.4 Comparison to other Grid projects

Direct comparison of quantitative metrics is difficult since so much depends on the hardware used and the state of the Internet at the time a trial is run – what matters is the relative overheads features included in different middleware introduce. Below are some similar activities and projects that provide a context in which this evaluation of GEMSS can reside.

3.4.1 Grid Resources for Industrial Applications (GRIA)

The GRIA project (GRIA) created a web services based Grid for industrial applications, including finite element analysis of structures (e.g. Dams) and 3D model rendering (e.g. for animated films). There are a number of similarities between GEMSS and GRIA due to the re-use of the GRIA accounts and process-based access control components in GEMSS. GRIA components were released as LGPL to facilitate this re-use.

The GRIA grid uses a three-step process, involving first a business step, then a resource allocation step and finally a job-handling step. The business step involves setting up an account and authorizing payment. Once payment is authorized jobs can be uploaded on a chosen service providers site and run. The resource allocation step allows CPU time, disk space etc. to be allocated for future jobs, within the context of an account to which it will be charged.

The users in GRIA request CPU time based on their judgement as to how long a job will take to finish. There are no application specific performance models in GRIA, although one of the partners investigated a neural network approach to server side performance modelling. The CPU time and resources used by a job comes out of the resource allocation previously made, until this resource allocation is used up.

The GRIA project supports HTTPS, WS Security, and process-based access control. There is a public key infrastructure but no end-to-end security protocols such as exists in GEMSS.

The GRIA accounts module provides a record of jobs run, and keeps a running bill that can generate a statement for the users at any time.

However, a exhaustive quantitative evaluation of the performance of GRIA was beyond the scope of the GEMSS project.

3.4.2 GridBench (CrossGrid project)

A set of tools called GridBench (G. Tsouloupas et al., 2003) has been developed for the Globus Grid toolset. These however do not port at all well to other Grids and hence are little use in evaluating web service based approached to the Grid.

The GridBench metrics focus on data throughput and execution time. They do not provide any metrics for security or robustness, the focus of the GEMSS evaluation.

3.4.3 GTA Globus ToolKit3 Evaluation Team

The LHC Computing Grid Project (OSAG) has spawned an activity called the GTA Globus ToolKit3 Evaluation Team (A. Demichev et al.). This team has evaluated Globus Grid's with and without various security options.

Turning on the GT3 security options resulted in a 10-fold decrease in throughput for small dummy service operations. Public key encryption using X.509 offers a 2-fold decrease in performance over basic encryption.

Globus does not offer end-to-end security and hence this aspect cannot be compared.

4 Summary and Conclusion

This document describes the final results of the evaluation and validation process that accompanies the GEMSS project. The testing methodology was defined, metrics identified, and a series of testbeds evaluated.

An evaluation of the software development was performed, requirements were compared to the final implementation and user and developer experience was collected in form of a questionaire.

All six medical applications are now Grid-enabled and turned into services, using the final phase I3 and I4 (technology demonstrator) versions of the GEMSS middleware.

All central goals defined in the initial requirements specification (D1.1) have been met. As shown in Table 1, a few middleware requirements were not fully implemented because they are only relevant in the exploitation phase.

In order to assess information about the run-time behaviour a variety of tests has been executed.

Stress tests were used to burden the middleware with a high load sufficient to break it. These tests revealed problems within the server side parts of the framework (mainly a bug in the external MySQL database software). All problems have been solved. Subsequent performance and robustness tests validated the GEMSS middleware design and demonstrated the high quality of the infrastructure.

A long term test of the middleware provided information about the standard use patterns and the efficiency and high reliability of the final GEMSS middleware. Negotiation and file transfer overheads were measured and found acceptable for the GEMSS applications. Test showed that the middleware is well suited for applications with run-times > 10 minutes such as the six GEMSS pilot applications.

Important feedback from the application developers revealed possibilities to improve the middleware:

- The certification process and Java keystore setup could be optimised.
- The Client installation of the middleware could be supported by a cross-platform GUI installer in order to hide the complexity of the installation process.
- A server side standardisation of the execution environment would help the application developers to easily distribute applications to different service providers.
- A tutorial describing the steps necessary to port a high performance application to the GEMSS infrastructure could be part of the final release.

At the time of completion of this deliverable (D1.3b was due at the end of November 2004) the next major steps towards deliverable D1.2c (due at the end of the project, February 2005) are:

- Deploy the client software at medical end user sites.
- Gather feedback from medical end users about the usability of the tools.

References

GEMSS Deliverable D 1.1 *Requirement Specification*

GEMSS Deliverable D 1.2b *Global System Design Revision*

GEMSS Deliverable D 1.3a *Midterm Evaluation and Validation*

GEMSS Deliverable D 2.2a *Methodology for Assessing Security*

GEMSS Deliverable D 2.2b *A global view on the European privacy legal framework regarding processing of patient s data considering GEMSS*

GEMSS Deliverable D 2.2c *COTS Security Technologies and Authorisation Services*

GEMSS Deliverable D 2.2d *A Global View on Contractual Terms for the Provision of GEMSS regarding European Law*

GEMSS Deliverable D 2.2e *Final Report on all Legal Issues related to GEMSS* (*due Feb. 2005*)

GEMSS Deliverable D 4.1 *Maxillo-facial Surgery Simulation*

GEMSS Deliverable D 4.2 *Quasi-Realtime Neurosurgery Support by Non-linear Image Registration*

GEMSS Deliverable D 4.3 *Cranial Radio-surgery Software*

GEMSS Deliverable D 4.4 *Inhaled Drug Delivery Simulation Service*

GEMSS Deliverable D 4.5 *Cardiovascular Software*

GEMSS Deliverable D 4.6 *Image Reconstruction Service*

GEMSS BugTracker, https://www.gemss.de/support/

M.A. Frumkin, R.F. Van der Wijngaart. *NAS Grid Benchmarks: A Tool for Grid Space Exploration*, Cluster Computing, 5(3), 2002

A. Snavely, G. Chun, H. Casanova, R. F. van der Wijngaart, and M. A. Frumkin, *Benchmarks for Grid computing: a review of ongoing efforts and future directions,* SIGMETRICS Perform. Eval. Rev., 30 (4), ACM Press 2003

GRIA, *Grid Resources for Industrial Applications*, http://www.gria.org

G. Tsouloupas, M. Dikaiakos., *GridBench: A Tool for Benchmarking Grids.* In 4th International Workshop on Grid Computing (Grid2003), Phoenix, Arizona, 17 November 2003

OSAG, *The LCG Grid Technology Area*, http://lcg.web.cern.ch/LCG/PEB/GTA/LCG_GTA_OSAG.htm

A. Demichev, M. Lamanna, A. Kryukov, R. Rocha, D. Foster·, C. Wangà, V. Kalyaev, *GTA Globus ToolKit3 Evaluation Team*, http://lcg.web.cern.ch/LCG/PEB/GTA/GTA-GT3testbed/GTA-GT3testbed.htm