



## QCDGrid2 System Risk Analysis

**Project Title:** QCDGrid2

**Document Title:** QCDGrid2 System Risk Analysis

**Document Identifier:** QCDGRID2-SWRA-1.0

**Document Filename:** qcdgrid2\_swra.doc

**Distribution Classification:** Commercial In Confidence

**Authorship:** Daragh J. Byrne (DJB)

**Approval List:** QCDgrid Development Team

**Distribution List:** UKQCD Collaboration

### Document History:

<i>Personnel</i>	<i>Date</i>	<i>Summary</i>	<i>Version</i>
DJB	27/OCT/04	Release Version	1.0

## Contents

<b>1</b>	<b>Introduction .....</b>	<b>2</b>
1.1	<b>The QCDgrid Project .....</b>	<b>2</b>
1.2	<b>Scope .....</b>	<b>2</b>
1.3	<b>Motivation .....</b>	<b>2</b>
1.4	<b>Objective .....</b>	<b>2</b>
<b>2</b>	<b>Risk Assessment .....</b>	<b>3</b>
<b>3</b>	<b>Risk list .....</b>	<b>4</b>
<b>4</b>	<b>Conclusions.....</b>	<b>30</b>
	<b>References .....</b>	<b>33</b>

# 1 Introduction

## 1.1 The QCDgrid Project

The UKQCD Collaboration [1] aims to “procure and jointly exploit computing facilities for lattice field theory calculations whose primary aim is to increase the predictive power of the Standard Model of elementary particle interactions through numerical simulation of Quantum Chromodynamics”. Such numerical simulations produce large amounts of data in the form of binary files. The first QCDgrid project developed a software suite running on dedicated hardware (the UKQCD Grid system) that allows UKQCD members to store and share data amongst one another in a straightforward and reliable manner. A graphical data browser tool was developed to facilitate ease of use. The software suite also allows users to initiate simulations and post-processing jobs on machines on the grid.

## 1.2 Scope

This risk analysis concerns the present and future state of the QCDgrid system in terms of software and hardware.

## 1.3 Motivation

The QCDgrid software and hardware system have been in successful operation since autumn 2002. As it stands the system is used by a user community at five geographically distributed sites and has coped well with the demands placed on it.

It is anticipated that this system will have to support the activities of the UKQCD community well into the future. The system will thus have to cope with changes in usage patterns and possibly changes in requirements. It is desirable to identify any potential problems that the system will have with these changing usage patterns before they arise, and outline strategies for coping with them.

## 1.4 Objective

This document aims to assess the ability of the current QCDgrid system to continue to support its user community into the future. The assessment takes the form of a risk analysis with risks enumerated in Section 3.

In line with the aims of the document, UKQCD have attempted to cover the complete space of risks that may impact on the system, including an evaluation of: usage, deployment, security, dependence on third-party software, reliability, and scalability. However, given the constraints on effort available to the project, the authors have exercised prudence in proposing actions to be undertaken. These proposals (documented in Section 4) are intended to minimise the likelihood and effect of individual risks, in a manner that is realistic within the project work plan.

## 2 Risk Assessment

The risks to the future operation of the QCDgrid system as they are understood at the present time are described in the following sections. These risks have been discovered during conversations with some end users, and by independent analysis by the software development team at EPCC.

Each risk is described under the following headings:

- Identifier in the form QR-N where N is an integer;
- Definition: a brief summary of the risk;
- Description: further detailed information about the nature of the risk;
- Scenario: a description of a scenario in which the risk affects the system (optional);
- Related risks: other risks that are related to this one;
- Likelihood: the likelihood of the risk occurring given current knowledge. High likelihood means that the risk is very likely to occur; moderate likelihood means that the risk is as likely to happen as not; low likelihood means that the risk is unlikely to happen. In many cases the likelihood can be affected by a number of factors. The factors affecting likelihood are discussed in this section;
- Impact: the likely effect the risk would have on the system, should it occur. High impact means that the risk has a severe effect on the system and will ideally be dealt with at some stage of the project – usually high impact will correlate with effort already having been scheduled in the project work plan [9]. Moderate impact means that it is not critical that we deal with this risk; but that non-critical problems could present themselves as a result of not dealing with it. Low impact means the risk affects the system in a cosmetic or otherwise trivial manner but that it would still be better to deal with the risk than not;
- Priority: how important it is to take action to control this risk. High priority means that effort will already have been scheduled in the work plan to deal with the risk. Moderate priority means that any effort that becomes available should be used to deal with the risk. Low priority means that the effort to deal with the risk need not be explicitly scheduled - that mitigation may happen in the background, or the risk may be monitored rather than dealt with;
- Control: actions that should be taken to reduce the likelihood of the risk occurring;
- Mitigation: action that should be taken in the event that the risk occurs.

### 3 Risk list

This table summarises the risks discussed in the following sections, ordered by priority.

Risk	Title	Priority	Page
QR-6	Increased load on system	High	13
QR-7	Insufficient test infrastructure	High	16
QR-10	Node unavailability	Moderate	21
QR-11	Security risks	Moderate	23
QR-12	Support for job submission insufficient	Moderate	24
QR-14	Data becomes corrupt	Moderate	26
QR-2	System component usability	Moderate	6
QR-3	Changes to hardware infrastructure	Moderate	8
QR-8	Third party software dependency	Moderate	17
QR-9	Technical expertise localisation	Moderate	20
QR-15	Change in requirements and bug reporting	Moderate	28
QR-13	Network problems	Moderate	25
QR-1	Ease of deployment and administration	Moderate	4
QR-4	Use of metadata system	Low	10
QR-5	Metadata schema changes/storage of other metadata	Low	11

Full details of each risk are presented in the following sections.

#### QR-1 Ease of deployment and administration

##### Definition

The build and deployment system may have or may develop usability difficulties. System administration of the software may have or may develop usability difficulties.

##### Description

The software may be at risk of failing any of the following usability requirements. Note that these are not formal requirements as collected from the user community; rather they are inferred from current usage patterns and problems.

- Members of the UKQCD collaboration who wish to build the QCDgrid storage node software on the supported platforms should be able to do so without encountering errors and without having cause to contact the development team. In the case where the build fails, detailed diagnostic information should be provided.
- Members of the UKQCD collaboration that wish to add or remove storage or compute nodes, or perform hardware upgrades on existing nodes should be able to do so without consulting the development team, and in a reasonable timeframe.
- Users should be able to install the client software without reference to the development team. This serves to reduce the support load on the QCDgrid software maintainers. It also serves to

increase confidence in the system within the community, as much of the impression a user has of a piece of software is formed during the install process.

- Installation of the metadata catalogue, control node and backup node should be possible in reasonable timescales.
- The QCDgrid software is covered by an open source license, which means it can be used within other projects external to the UKQCD Grid. It is important that the installation and maintenance process be straightforward (with reference to the above criteria) to enable such users to get on with their projects.
- It should be possible to perform routine maintenance tasks in an intuitive and straightforward manner.
- The system should provide the user with appropriate information to diagnose installation and maintenance problems when they occur.
- Installation and administration of software developed in the future should meet similar usability requirements.

### **Related Risks**

System component usability [QR-2]; Technical expertise localization [QR-9], Changes to hardware infrastructure [QR-3]

### **Likelihood**

Moderate.

Two defects related to usability have been entered in the NeSCForge tracker in the last two years. This is considered to be a low number of usability defects. At present, one of these defects still has to be resolved.

In earlier versions of the software, problems were experienced while installing the Globus toolkit (which is a prerequisite of the software). Use of the Virtual Data Toolkit (VDT) [2] to install Globus has practically eliminated this problem, and its use will continue to be recommended. This will need to be re-evaluated should the software be rewritten using middleware other than Globus 2 (see QR-8).

A well maintained guide to the installation process has been produced by the QCDgrid team [3]. A number of ways to improve this document have been suggested by the community via the NeSCForge site [4], and these suggestions will be incorporated into the document, but overall it has been found to be satisfactory.

Currently the most common administrative tasks are temporarily disabling a node and permanently removing a node. These are achieved using a simple command line executable, and no major usability problems have been reported. Administration of the central node has also proceeded without presenting any usability problems. This has included moving the central node from a test machine to its current permanent residence.

It may be concluded that at present the installation system is satisfactory, and the maintenance tasks relating to the QCDgrid software are easily achieved. Provided major changes maintain this ease of use, the likelihood of this risk affecting the current software distribution is low.

There is a moderate likelihood of this risk affecting software that will be developed in the future, particularly the ILDG web services, since the installation method and administrative requirements are not yet known.

**Impact**

High.

A user's first experience with a piece of software (usually the installation process) colours their impression of the software as a whole. The long term experience is shaped by how easy the software is to administer. There is a danger that users will not use the software fully, become frustrated with it, or lose confidence in the quality of the software if usability problems occur. This is more relevant to individuals that have yet to install the software than those who have already done so.

**Priority**

Low.

Ensuring the install process is as painless as possible does not require that specific effort be scheduled; rather that usability issues are considered as part of the overall software design and maintenance processes.

**Control**

- Conduct review of software installation process, This will be done when setting up the test grid for stress testing (see QR-6);
- Design installation scripts, makefiles, etc. to meet usability requirements;
- Ensure installation and maintenance guides exist for all software, including troubleshooting information. Keep these guides up to date with respect to new releases of the software;
- Monitor and incorporate feedback regarding usability from the user community into maintenance plans;
- Where possible, use existing solutions to solve usability problems (an example being the use of VDT to install Globus);
- Assess information provided to users on failure of components and see if it is useful in diagnosis of problems;
- Ensure usability guides the design of new software.

**Mitigation**

Should usability of the install system or maintenance of the software become a large concern, effort will be allocated from the maintenance budget (WP 1.4) to rectify the situation.

**QR-2 System component usability****Definition**

The user-facing system components may have or develop usability problems.

**Description**

The QCGrid system's functionality is exposed through a number of user-facing components such as command line tools and GUI programmes. There is a risk that these user-facing portions of the software used on a daily basis (e.g. command line clients, GUI tools) fail some or all of the following usability criteria:

- New users of the software within the UKQCD community should be able to use the software using only the documentation provided and without recourse to explicit technical support by the development team;
- Users of the software within external projects should be able to use it without recourse to explicit technical support by the development team;
- Functionality expected by the user is where they think it is – i.e. the software interfaces are intuitive;
- Interfaces provide all functionality expected by the user;
- Software should provide appropriate and useful diagnostic information when problems occur.

These requirements also apply to software developed in the future.

### **Related Risks**

Ease of deployment and administration [QR-1], Insufficient test infrastructure [QR-7].

### **Likelihood**

Low/moderate.

Most of the user-facing QCDgrid functionality, with the exception of the browser, is implemented as a set of command line clients. This interface seems to be well accepted by the user community, and thus is considered to have good usability. This is evidenced by the fact that only two defects relating to usability have been filed on the NeSCForge project tracker in the last two years.

The data browser has not been in operation in the field for enough time to make judgments on its overall usability. If the data browser becomes the primary means of interaction with the grid, further assessments of its usability need to be made. The data browser allows queries to be run on metadata using arbitrary schemas. The controls on the browser are generated dynamically by referencing the schema. There is a moderate likelihood that this automatic generation could result in unusable or difficult to use interfaces.

This risk carries a low likelihood, provided the interfaces to the software are maintained as they are. There is a moderate likelihood that the software developed in the future will have usability problems.

There is a moderate likelihood that software in the future will have usability problems. This should be considered when designing the software.

### **Impact**

Moderate/high.

The usability of the software has a high impact on the success of the software, both inside and outside the UKQCD community, as it is a primary factor influencing user satisfaction. Care should be taken to ensure the level of usability is maintained.

If software developed in the future suffers from usability problems, negative impact on user satisfaction and uptake of the software can be expected.

### **Priority**

Moderate.

It is important that the usability of the software is maintained. Priority should be given to ensure that the usability of the software remains high, and that problems in the current software be dealt with as they emerge.

As regards the development of new software in the future, consideration should be given to usability issues within the design, testing and reviewing stages of the project, although no specific effort needs to be scheduled.

### **Control**

- Avoid changes to the command line and GUI interfaces where possible;
- Document and publicise any changes to the interfaces should they occur;
- Maintain backward compatibility between versions of tools. In the case that new command line tools are developed, the switches used should be consistent with those already in use. The policy of allowing the user to print out the correct usage for a command line should be adhered to. Documentation for the command line tools should be maintained;
- Test data browser with every schema it is expected to support;
- Collect usability feedback internally and from beta testers. Monitor use of data browser by community to ensure its suitability for use. Monitor usability of command line clients. Consider usability related comments from the community when deciding on features for a release;
- Assess behaviour of the software interfaces under failure conditions, specifically with respect to diagnostic messages produced.

### **Mitigation**

Should usability of the install system or maintenance of the software become a large concern, effort will be allocated from the maintenance budget (WP 1.4) to rectify the situation.

## **QR-3 Changes to hardware infrastructure**

### **Definition**

Changes to the hardware infrastructure, including the capacity and distribution of storage space, replacement of existing nodes or the addition of new nodes (either storage or processing), may affect the system adversely, or the system may be found to be defective under these changes.

### **Description**

This risk pertains to the data grid and job submission software. The following are examples of hardware infrastructure changes that could cause problems:

- Installation and maintenance of the software for users at new sites;
- Transfer of a logical node to new physical hardware;
- Integration of new processing node hardware with the grid;
- Addition or removal of storage nodes. This may result in an unequal distribution of storage space – one node may contain more storage space than all the other nodes put together, which means replication of some files may not occur;
- Addition of nodes (of any type). This may cause software scalability problems;

- Removal of storage nodes. This results in a loss of storage space, plus the instigation of a large scale replication effort;
- Introduction of new software components, such as web services, that require the addition of hardware to the grid.

This risk essentially means that any of the related risks could happen due to the hardware infrastructure of the grid being changed.

### **Related Risks**

Increased load on system [QR-6], Ease of deployment and administration [QR-1], System component usability [QR-2], Node unavailability [QR-10].

### **Likelihood**

Low/moderate.

According to Richard Kenway, “the UK part of QCDgrid will I think be limited to Edinburgh, Glasgow, Liverpool, RAL, Southampton and Swansea”. Therefore it looks likely that any hardware changes will involve replacement of machines, or addition of storage space to existing machines. At present, we are not aware of any plans to remove machines from the grid, though this possibility should not be excluded from future planning.

Given the current proposal to expose each grid in the ILDG collaboration via the use of a web service, it might be necessary to introduce new hardware to the grid to stage the web service. The likelihood of this is difficult to estimate until the web service is designed. It is probable however that the web service could be run within the Tomcat server currently hosting the metadata catalogue on the central node. However it is likely that a staging node for serving files will be required.

See also the likelihood section of each of the related risks.

### **Impact**

High.

If the software fails to scale to multiple nodes the requirements have not been met, and effort will need to be expended to resolve the problem. The scalability testing phase of the project should give clearer insight as to the likelihood of this happening.

See the impact sections of the related risks.

### **Priority**

Moderate.

### **Control**

This is a high impact risk, but controls scheduled for other risks (especially [QR-6]) should act overall as a control on this risk. Other possible controls include:

- Use the test grid to simulate a large number of nodes;

- Use the test grid to simulate the case where one node has a much larger storage space than all of the others combined;
- Monitor the distribution of storage space on the grid;
- Ensure that all consortium members are aware of planned changes to hardware via announcements on the mailing list or at the video conferences;
- Ensure technical support by the development team is available when we know new machines are being added to the grid.

**Mitigation**

See the mitigation sections of the related risks.

**QR-4 Use of metadata system****Definition**

When the metadata system starts to be used, the system may be adversely affected or may be found to be defective.

**Description**

At the moment most data is stored on the Grid without metadata. System testing has shown the metadata storage system, and the browser, to work; however, other defects may be noticed when the system comes into widespread use by a large number of users.

Ideally users will store metadata with every piece of data on the grid. This will enable other users to search for data, and will be extremely useful when the ILDG comes online for global collaborators. It is necessary to ensure that users are aware of this functionality and to ensure that they are actually using it.

There is a risk that the metadata storage system fails to scale with the number of users or volume of data. The details of this risk are covered in [QR-6].

**Related risks**

Increased load on system [QR-6].

**Likelihood**

Low/moderate.

As stated, system testing has shown that the system meets its original requirements of allowing users to build queries and locate logical files on the grid. Further in-the-field use may illuminate other defects, but functional defects are less likely to be found. Furthermore, the total amount of data stored in the metadata catalogue is significantly less than the amount of data on the grid, so it is unlikely to cause problems.

There is a moderate risk that users will have to be encouraged to use the metadata system, as at present it is not being used to its full potential.

**Impact**

Moderate.

If the functionality provided by the metadata catalogue is not used, this is a reduction in the overall usefulness of the system. Essentially redundant software will have been developed.

**Priority**

Low.

Effort has not been allocated for further functional testing of the metadata system. Defects that appear in the future will be dealt with in normal maintenance releases of the software. However, the load and scalability characteristics of the metadata system have been assigned a high priority as part of the overall load testing plan (see [QR-6]).

Ensuring the system is used is assigned a slightly higher priority. The system is much more useful if every piece of data has metadata associated with it – this will be especially true when the ILDG comes online.

**Control**

- React to functional/usability defects as they come up;
- Deal with scalability aspects of the system as part of the overall scalability test plan;
- Build awareness of the functionality and importance of the metadata functionality within the community;
- Investigate feasibility of designing and implementing policies that would ensure metadata system is used, such as forcing a user to submit metadata with pieces of data – this would involve modifying the existing data grid software;
- Publicise the metadata features of the software via email lists, video conferences etc;
- Ensure usability of UIs to metadata functionality to help take up of metadata by community.

**Mitigation**

- Increase awareness of the importance of using the metadata system by emailing users and talking to technical contacts at each centre;
- Enforce use of system in software, by forcing users to submit metadata with data.

**QR-5 Metadata schema changes/storage of other metadata****Definition**

The QCDML metadata schema may change. Metadata using a schema other than QCDML will be stored in the system.

**Description**

At present the metadata catalogue is intended to hold information describing the binary files stored on the grid. In general these binary files contain the results of simulations and results of other processing. The metadata catalogue holds XML documents that should conform to the QCDML schema.

While it is currently stable, it is possible that the QCDML schema may change in the future. Although the software has been designed to cope with a changing schema, it may be affected in some unanticipated manner.

It is also anticipated that users will start to store other types of binary files on the data grid e.g. executables that are used to generate data, or source code files. In this case, it may be desirable to store metadata about these files in the metadata catalogue. There is a risk that this will be done in an arbitrary, uncontrolled manner, resulting in differences in metadata schema for similar files.

**Related Risks**

System component usability [QR-2].

**Likelihood**

High.

QCDML is currently at version 1.1. This release is intended to be relatively stable; however, as the metadata facilities of the grid are used more frequently, deficiencies will likely be found. It is anticipated that only minor changes will occur in the short term. Should major problems be found in the longer term, more significant rewrites could be anticipated.

Craig McNeile has suggested that storing executable files on the grid, with their metadata, as well as other types of XML based data files, is desirable.

**Impact**

Low.

In terms of the software, very little would have to be rewritten to deal with changes to the metadata schema. The browser has been written to adapt its appearance to changes in the schema. The only issue with the browser is its usability with new schema (see [QR-2]). There may be an issue with users having to change the schema against which the browser is searching a number of times – hence having to perform multiple searches each time.

Most XML database implementations, including the one used in QCDGrid (eXist), do not take schema into account when storing XML data. Data is stored as collections of documents of arbitrary type. The structure of the database will not have to be adjusted if the QCDML schema changes. This means that users can submit any XML file to the metadata catalogue without changes being made to the system. This may result in the system being used in ways that were not intended, which may lead to a difference in the perceived requirements of the system between the developers and the users. This situation could lead to a certain amount of confusion in the community as to what can and cannot be done with the system.

It is very likely that some non-QCDML metadata files will be stored on the grid as if they were regular files, bypassing the metadata catalogue altogether. There is no direct impact on the system when it is used like this – after all, it can be used to store arbitrary files. In this case, it is up to the users to keep track of this data, and to ensure that appropriate schema and tools are available to users who wish to work with the data.

**Priority**

Low.

As stated there is little risk of the QCDML schema changing dramatically in the future. However there is a suggestion that the metadata catalogue be used for at least one type of metadata other than

QCDML (see comment above about storing executables on the grid). The priority for developing a policy on this is moderate.

### **Control**

- Gather formal requirements for the type of metadata that may have to be stored in the future;
- Define a usage policy for the metadata system, describing what is acceptable usage and correct forms of metadata;
- Possibly re-engineer the data store so that related pieces of metadata are stored in separate collections;
- Define a process for allowing the ability to store a new type of metadata and for alerting the community to this desire;
- Ensure members of the user community keep track of non-standard metadata that is placed on the grid, and make any schema or tools used with this metadata available;
- Investigate the feasibility of enforcing usage policies in the software through the metadata submission software.

### **Mitigation**

- If the QCDML schema changes no technical action is necessary;
- If the way the metadata system is used changes, consult the community as to which of the controls (above) would be desirable to implement.

## **QR-6 Increased load on system**

### **Definition**

The load placed on the system will increase in the future, which may adversely affect the performance, reliability and scalability of the system or show it to be defective in some other manner.

### **Description**

The following are the basic use cases currently supported by the QCDgrid software:

- Submit jobs that produce data (e.g. simulations that produce configurations);
- Submit jobs that act on data already present on the grid (e.g. some processing involving previously generated configurations);
- Store data file/files on the grid;
- Store metadata file/files on the grid;
- Search metadata catalogue;
- Retrieve data file/files from the grid;
- Retrieve metadata from the grid;
- Delete data files from the grid;
- Delete metadata files from the grid.

Many combinations of the basic use cases are possible; for an example see the scenario below.

The user-initiated load on the system comes from initiating jobs, transferring data from and to users (i.e. reading and writing data) and querying the metadata and replica catalogues. Background load comes from the bookkeeping and replication activities of the central node. Of the user-initiated load, it

is envisaged that writing data to the grid will occur considerably less than reading data from the grid, as data is only written once. It is not yet known what level of load on the grid compute nodes will be generated by job submission.

Currently the data management system operates under low to moderate loads. The job submission system is not in widespread use just yet (it is anticipated that all jobs will be submitted via this system); therefore the loads due to this component are low. The system performs comfortably at its current loads. The risk is that, as the load placed on the system increases, it will fail to scale. There are a number of potential causes of such failure:

- The network bandwidth for doing the data transfer is insufficient;
- The QCDgrid software does not scale well when large amounts of data are placed on the grid;
- The QCDgrid software does not scale with the number of requests it has to deal with;
- The QCDgrid software puts unnecessary strain on the system due to its background activities;
- Third party software (such as Globus) fails to cope with the increased load (see QR-8).

### **Scenario**

A typical scenario is for a user to submit a job to a large processing node; e.g. a simulation. The results of this job (several files) are stored to the grid with logical filenames (LFNs). The user wants to do some post-simulation work using these results; another job is submitted to the grid using the LFNs as input parameters. The secondary job starts, and copies the relevant file or files from the grid to the node on which it is executing. The secondary job produces more data, which is stored on the grid. Some time later (e.g. the following week), the user wishes to obtain a copy of some of the data and downloads it from the grid to their local machine, using a metadata search to remember details of the data.

### **Related Risks**

Insufficient test infrastructure [QR-7], Changes to hardware infrastructure [QR-3], Network problems [QR-13].

### **Likelihood**

High.

Two major milestones for increased system load have been identified. The first is the appearance of the QCDOC machine. This machine will allow the production of much more data than is currently being produced. Large increases in the number of jobs being submitted to the grid, as well as the amount of data being transferred around the grid, are also expected when this machine goes live. This is expected to happen in October or November of 2004.

The second milestone for increased load is the advent of the ILDG. In this scenario, increased read activity is expected to occur as data is shared with other ILDG collaborators. This is expected to happen during 2005.

These events are definite, and will result in an increased load on the system.

Although it is unlikely that the dedicated QCDgrid hardware will expand beyond the current sites at Edinburgh, Glasgow, Liverpool, RAL, Southampton and Swansea, there is a possibility that the QCDgrid hardware and the GridPP/LHC grid may be integrated at some undetermined time in the future. This may result in further indirect load being placed on the system.

**Impact**

High.

We need to be confident that the system can operate at its expected loads. The loads under which each component of the system - software, network, or hardware - may fail are not yet known. The increase in load due to each new development has yet to be estimated. This is a high impact risk to the whole QCDgrid project, and effort has been scheduled to deal with it explicitly. Effects could include catastrophic system failure, data loss, and a lot of angry users, so care needs to be taken to control this risk.

**Priority**

High.

Effort has already been scheduled in the project work plan to control this risk.

**Control**

Effort has been allocated to develop a load testing suite for the QCDgrid system. Since the system load is likely to vary widely, with the types of jobs being run and the amount of data being moved around being extremely hard to predict, it is practically impossible to assess a “typical” system load. Testing at worst-case loads is anticipated.

Actions necessary to develop this test suite include:

- Speak to the user community in order to assess the likely loads on the system;
- Quantify the number of users; the maximum numbers of jobs they expect to submit to the grid in a given period of time; the maximum amount of data they wish to transfer weekly;
- From this information, establish a set of “worst case” loads upon the system for each of the use cases defined above – estimating “typical” loads is difficult, as usage patterns change from hour to hour;
- Define a set of formal load requirements on the QCDgrid system – possibly using the “worst case” loads to help define acceptance criteria. Obtain feedback from the community about these load requirements.

It is proposed that the test suite be flexible enough to test each feature of the system at any load (in an easily configurable manner). This is to allow routine testing of the grid before each ramping up in load.

Testing should occur in two phases. The first phase involves running the suite on a local test grid under realistic stressful conditions, to investigate possible failure modes. There are limitations in replicating the actual state of the production grid on the test grid, so the second phase may involve running the same tests on a subset of the production grid nodes, using carefully marked dummy data.

The community should be monitored in order to be aware of any potential increases in the load on the system, such as the addition of new machines. This could possibly include producing usage statistics on a periodic basis (e.g. monthly).

**Mitigation**

If the system proves not to cope with the load put upon it, effort will have to be scheduled to track down and remedy the causes. Solutions could involve using policies that restrict hours during which sets of users can use the grid, or technical solutions such as rewriting parts of the software.

## **QR-7 Insufficient test infrastructure**

### **Definition**

The test infrastructure may prove to be insufficient to rapidly test changes to the system.

### **Description**

At present, most testing of the QCDgrid system is performed manually. Given that the system will evolve over the coming years, both in terms of functionality and user community, it would be useful to automate some of the testing procedures to save on testing effort and prevent regression scenarios.

### **Scenario**

A change is made to the implementation of the “put-file-on-grid” executable (with no functional changes). At present, the change would be tested manually, i.e. running the command and checking each node to see if the file is indeed uploaded somewhere on the grid. With a dedicated testing suite, it would be possible to run a simple script to test that the program still operates as expected, as well as checking that the change has not affected any other part of the code, in one simple step.

### **Related Risks**

Increased load on system [QR-6], System component usability [QR-2], Ease of installation and administration [QR-1].

### **Likelihood**

High.

A test script for testing the installation of Globus exists. Manual testing procedures exist – when changes are made the features are tested manually. Acceptance and beta testing are well established, with a core of system users providing early feedback on software releases.

There are, however, areas that could perhaps be improved. A thorough unit testing framework would be a useful addition; however such tests are difficult to write in a distributed environment that uses a large amount of middleware.

### **Impact**

Moderate.

This risk affects future maintenance and development of the system.

The data grid software has been demonstrated to be functionally stable. The job submission software has not been as extensively tested in the field as the data grid software; it is expected that this will change when QCDOC comes on line. The same comment applies to the metadata browser.

### **Priority**

Moderate.

Beta and acceptance testing are implemented in a satisfactory manner, with a number of users in the community willing to provide feedback on early releases. Priority should be given to ensuring that this remains the case.

Implementing unit test suites for existing code is low priority. Future development should involve dedicated unit testing where practical, for example using JUnit [5] to test Java code.

Automated system testing is of a high priority and will definitely be addressed. Effort has been allocated within the project plan to address this.

Installation testing is easily implemented, based on existing test scripts. The priority for this is low, as the installation process currently performs well.

### **Control**

- Implement a test suite;
- Give priority to developing adequate test suites to future software, by using “test-driven development” (of which members of the development team have experience);

### **Mitigation**

Should testing become a large issue, further effort will be allocated to deal with needs as they arise.

## **QR-8 Third party software dependency**

### **Definition**

The QCDgrid software relies heavily on third party components, such as Globus. If it becomes necessary to stop using these components, parts of the system may have to be rewritten.

### **Description/Scenarios**

The software currently relies on a number of third party components:

- Globus Toolkit 2.x (for the data grid and the job submission software);
- GridPP Virtual Organisations;
- eXist XML database.

These may become unusable for a number of reasons.

The first reason is technical. As the system evolves and the load placed upon it increases, it may be found that problems exist with the scalability of the third party components. Load testing may also reveal this to be the case. Security flaws may be found. A new product that is somehow “better” may become available.

The second reason is support related – the versions of the third party components used may become unsupported. For example, the most current version of Globus is 3.9, with the release of 4.0 imminent. It is possible that the version of Globus used (2.4) may not be supported into the future. It is usually not desirable to run a production system using unsupported components. However, it should be taken into account that it is also unwise to re-write a working system.

Future software is also likely to depend on third party components as well, specifically Web Services environments such as Axis. This dependence carries the same risks.

### **Related risks**

Increased load on system [QR-6].

### **Likelihood**

At present, the risk has a low to moderate likelihood of occurring.

The technical quality of the software has so far been satisfactory. Failures due to loads have not yet occurred, but see [QR-6]. Globus has been found to operate correctly. Globus version 2.x has been deployed successfully in many grid environments worldwide. The metadata catalogue part of the data grid also appears to function well at present loads. Not much is known at present about the ability of eXist to scale; however, load testing should illuminate any deficiencies in this regard.

As regards support, it is unclear whether Globus 2 will continue to be developed much beyond its current state. In functional terms, this is fine as it currently provides the functionality needed. If a serious defect were to be found after it stopped being supported then it is very likely that the software would need to be rewritten, or the defect remedied by the development team.

### **Impact**

High.

The impact of this risk is different for each component of the software. In the data grid software, Globus functionality is well encapsulated in functions in a number of source code files. These files include

- `gridftp.c`
- `job.c`
- `node.c`
- `replica.c`
- `client.c`

The rest of the QCDgrid software functionality is built upon code in these files. A rewrite of these functions using appropriate middleware should solve the problem, if the middleware provides the following functionality:

- Replica catalogue;
- Job submission;
- Cross-domain security.

There is an impact on the security system currently in use. The Globus GSI system is used along with GridPP Virtual Organisations. If the selected middleware replacement does not support this mechanism, it is possible that the whole software suite will need to be rewritten. This has severe impacts on both effort available within the project, and possibly usability. However, it is likely that a solution based on the certificates issued by the UK e-Science Certificate Authority could be developed.

The Java data browser relies on a library compiled from the above C files for its grid functionality. The library is bound to using the Java Native Interface (JNI). The Java code should continue to work as normal with the updated library. If the library is rewritten in a manner that exports the same functions, there will be little impact on the Java code. Should the core grid software be re-architected however, it is likely that JNI bindings would need to be rewritten.

The grid parts of the software could be re-written using a number of options. These options may have to support an alternative to the current security set-up, which is dependant heavily on the current middleware. The options include:

- Later versions of Globus, such as version 3.0 (which implements the OGSi standard [9]) and version 4.0 (which will implement the WSRF standard [11]);
- Web services technologies (such as Axis, though the security infrastructure would have to be re-written);
- Unicore [12];
- Grid Application Toolkit (GAT) [13], an API for abstracting grid middleware. GAT itself depends on further middleware solutions such as those mentioned above, but provides an abstraction layer that would insulate the QCDGrid software from changes to the underlying middleware;
- An ad-hoc solution based on sockets.

Alternative XML databases include

- Xindice [14], another open source XML database that has the advantage of supporting the XMLDB API currently used by the software; however performance has been an issue in the past;
- Tamino, a commercial XML database product from Software AG. This has the disadvantage of not supporting the XMLDB API, and would have to be paid for;
- An ad-hoc solution (undesirable as requires XML database implementation expertise and extra effort).

### **Priority**

Moderate.

There are no plans immediately to rewrite large portions of the software. However, two software reviews have been scheduled, where the middleware used will be examined. Software will only be rewritten if these reviews find it necessary. The priority will be reassessed at the time of the software reviews.

### **Control**

- The evolution of Globus and eXist will be tracked by subscribing to relevant mailing lists;
- Regular software reviews have been scheduled in the project work plan to see if a rewrite is (or will become) necessary;
- In the event of a rewrite the user interfaces to the software will be kept as close as possible to the current interfaces in order to minimize disruption to users. Furthermore, efforts will be made during this rewrite to isolate the third party code as much as possible, in case further rewrites become necessary.

### **Mitigation**

- Allocate effort for necessary rewrites, possibly out of the maintenance budget.

## QR-9 Technical expertise localization

### Definition

Technical expertise about the software system becomes too localized, putting unnecessary strain on a small number of experts.

### Description

At present many support requests and defect reports are handled directly by the development team. These requests have included: installation, administration and defects. These requests are handled by the development team because it is perceived that the expertise around the software resides with the developers.

It is desirable to avoid knowledge localization for the following reasons:

- An undue burden of effort is placed on the experts;
- Knowledge may be lost if people leave the project;
- Response times to problems can be lowered if more people are available to deal with them.

This situation can be avoided if the knowledge to deal with common queries exists within the community, and the software is well documented.

### Related risks

Ease of deployment and administration [QR-1], System component usability [QR-2].

### Likelihood

High.

Nearly all support requests around the software are handled by the development team.

### Impact

Moderate.

The only ill effects are diversion of effort from development to support. It would obviously be beneficial to avoid this.

### Priority

Moderate.

Effort has been budgeted for support and maintenance within the QCDgrid project. It is possible that some of this effort could be allocated to implementing some of the controls below.

### Control

- Ensure the information exists, such as user manuals, that will allow people to help themselves when using the software;

- Identify common situations that are often interpreted as defects by users but are actually caused by misunderstanding;
- Ensure that users (new and old) are aware of the resources available to help with the software, perhaps by direct dissemination of the resources via the UKQCD mailing list;
- Train people at other sites to take over support tasks;
- Set up a query distribution system so that queries are handled by multiple individuals at different sites in a round-robin fashion;
- Set up the NeSCForge support tracker to email all users of the grid – this encourages all users to know about the issues, and increases the numbers of those who can answer queries;
- Encourage a culture of relying on the community as a whole rather than directing every support request to developers, perhaps by better use of the mailing list;
- No matter what measures are adopted, ensure they have the support of the community, as alienating users is not desirable.

### **Mitigation**

- Allocate time from the maintenance budget to deal with the problem.
- Mass mail all users to make aware of the correct procedures when having difficulties.

## **QR-10 Node unavailability**

### **Definition**

A node or several nodes in the system may become unavailable, through failure or retirement. This may result in system functionality becoming unavailable, or data loss or corruption.

### **Description**

The QCDgrid system consists of a number of nodes, each having a different purpose: overall control of the system; storage of the data; execution of jobs. One or more of these nodes may become unavailable for a period of time, or permanently. One of the requirements of the system is that data is stored reliably (in multiple locations). There is a further requirement that the job submission system behaves sensibly in conditions where nodes are unavailable.

There are a number of possible ways in which node unavailability might affect the system:

- The central node might fail or become otherwise unavailable, in which case the system becomes read-only; there is a possibility that this might cause jobs running on compute nodes to hang as they attempt to write data to the system, wasting compute resources;
- The backup node may fail or become otherwise unavailable, in which case the data storage system becomes completely unavailable; the risk of interfering with running jobs applies in this situation as well;
- Separate components of the central node/backup node (i.e. the replica catalogue and the XML database that stores the metadata) may fail or otherwise become unavailable, meaning no access to the functionality dependent on them (e.g. searching the metadata catalogue); also, since these components are hosted on the central node, they will all become unavailable should it be taken off-line;
- Storage nodes may fail or become otherwise unavailable. New data on these storage nodes (i.e. data that has been submitted to the grid but has not yet been replicated and moved to its final location) will be lost unless the storage nodes are restored. If multiple storage nodes become unavailable, data that is replicated only on those nodes will become unavailable;

- Storage nodes may be retired permanently. At present, this will result in any data in the process of being put on the system (i.e. data in the “inbox” which has not yet been moved to its permanent home and replicated) being lost; however, other data stored on the node will be replicated by the system;
- Storage node failure may cause jobs executing to hang as they attempt to write to/read from that node;
- Compute nodes may fail meaning jobs may die and cannot be submitted to that node.

Nodes may become unintentionally unavailable, for example through hardware or power failure; or intentionally unavailable through scheduled maintenance or retirement. In general, the system takes no special actions when nodes are found to be unavailable. The system can currently deal gracefully only with the failure of a storage node on which there is no new data waiting to be replicated. It periodically checks all nodes to see if they are still operational. If not, replication of the data stored on that node occurs.

### **Related risks**

Increased load on system [QR-6]; Network problems [QR-13]; Data becomes corrupt [QR-14].

### **Likelihood**

Moderate/high.

It is almost certain that all possible modes of node unavailability will occur throughout the lifetime of the system, though it is hoped that the unscheduled failures will occur infrequently and for hardware reasons rather than malfunctioning software. Permanent retirement of storage nodes is not likely to happen.

### **Impact**

High.

The exact impact of each mode of unavailability has been described above. Should higher reliability be required, it may be required to re-architect the system, which could involve considerable effort.

### **Priority**

Moderate.

The reliability requirements of the system have not been defined. If it is found that the system (or any component) needs to be more reliable than at present, the priority to control this risk will be raised.

### **Control**

- Investigate the degree of reliability required for the central node, as this is the single point of failure of the data grid system (although it can be backed up). Investigate viability of distributing the catalogues over multiple nodes. Investigate possibility of running a control thread on multiple nodes;
- Identify whether lack of write access will in fact cause jobs writing to the data grid to hang – this can be integrated into the test plan;
- Identify whether continuous monitoring of each node in the system for problems is feasible, to avoid unscheduled downtime – perhaps integrating an “early warning system” when a node is in trouble;

- Modify the system so that when a storage node is retired it moves the data in its “new” folder onto another node on the grid;
- Provide documentation to the user community about what they should do if a node in their control goes wrong;
- Investigate whether intentional node downtimes could be coordinated;
- Test each failure mode to ensure that the system responds sensibly. This could be done during the stress testing phase.

**Mitigation**

- Reaction depends on the type of node that becomes unavailable, but should involve keeping the user community informed about what is going on.

**QR-11 Security risks****Definition**

There may be security risks to the system as a direct result of code written by UKQCD. Future software, especially ILDG work, needs to be written in a secure manner.

**Description**

This risk only considers code written by UKQCD; the QCDgrid system relies mainly on third party middleware for its security features (i.e. Globus security). The system uses security features (such as Globus security) that have been well accepted by the community, and this risk specifically does not include security provided by the middleware. The consequences of defective middleware have been examined in [QR-8]. This risk deals only with QCDgrid code written by UKQCD.

While a thorough software quality process has been followed, no explicit security reviews or security testing have been carried out. There is a risk that the C code contains security flaws such as buffer overflows that could leave the whole system open to attackers, should they first break through Globus security. There is a risk that the Java code could suffer from various security vulnerabilities in the same circumstances.

When authoring software that exposes the features of the system to the outside world in the future, such as ILDG web services, the attack surface for the system is increased. There is a risk that the chosen architecture for future software is not secure.

**Related risks**

Third party software dependency [QR-8].

**Likelihood**

Moderate.

**Impact**

Potentially high, but generally unknown.

Impact could range from denial of service, through to gaining control of nodes, through to corruption or deletion of data. Whether individuals are interested in attacking the system is unknown, though any

machine on the internet is vulnerable to attack. A thorough security audit, including threat assessment, would be required to assess the impact on the system.

**Priority**

Moderate.

Attackers do not need specific motivation to attack particular systems. The damage done to compromised systems tends to be reckless – e.g. deletion of data. Since maintaining the integrity of data is one of the primary reasons for building the QCDGrid system, security should be high on our list of priorities.

**Control**

- Perform a full security audit of the code as it stands, using tools such as Flawfinder [6] or RATS [7];
- Investigate whether other security auditing tools are available that could help track down vulnerabilities. Where such tools exist, use them to audit our code base;
- Use threat modelling [8] or some other formal security method when developing new software.

**Mitigation**

- Schedule effort from the maintenance budget to deal with any security problems found;
- Keep the user community informed of any problems.

## **QR-12      Support for job submission insufficient**

**Definition**

The job submission software supports only a limited set of batch submission software. This may need to be extended, requiring effort to be expended. Additionally, defects may be found in the job submission system.

**Scenario**

A new compute node is added to the grid that uses a different batch submission system from those supported by the software. The software is unable to interact with this node.

**Related risks**

None.

**Likelihood**

High.

At present the system only supports PBS. It is highly likely that other batch submission systems will need to be supported as nodes are added.

At least one bug relating to the job submission software has been noticed since its release to the community. It is probable that more bugs will be noticed, particularly when new code is added to the system.

**Impact**

Moderate.

This will require that effort be expended. However, it will probably be possible to leverage support within the Globus toolkit for other batch systems.

**Priority**

Moderate.

Effort from the maintenance budget will be used to further develop the job submission system to deal with other batch submission systems.

**Control**

- Identify batch submission systems that need to be supported;
- Allocate effort from the maintenance budget to implement support for these systems.

**Mitigation**

See control.

**QR-13 Network problems****Definition**

Network problems may cause difficulties for the system, especially for the software components.

**Description**

The QCDgrid software uses the network in the following manners:

- Perform system control tasks such as replication and data integrity checking – these rely on the network for communication between nodes;
- Perform functionality related tasks such as data transfer and job submission via Globus.

There are three situations where the software may fail because of network problems:

- The software is functioning as efficiently as possible, but the network is insufficient;
- The software is functioning inefficiently causing problems with the network;
- The software is not 100% efficient but the fact that the network has excess capacity masks this – however the lack of efficiency may show up in the future.

In some cases the problem is with the network itself; in other cases the software is using the network inefficiently. In the event that the software is found to be causing problems with the network, effort will need to be expended to track and fix the problem.

**Related risks**

Node unavailability [QR-10], Increased load on system [QR-6].

**Likelihood**

Moderate.

Most nodes are at the end of high bandwidth connections to the internet. There has been at least one instance where lack of bandwidth caused problems with a node (the Swansea node); however in that case the network has been upgraded.

There is the chance that the messages sent by the control node cause an overhead that interferes with the operation of the network, though this is not likely as the messages sent are generally small.

The most likely problems are that the network will encounter difficulties with extremely large datasets, or the network will otherwise be unable to deal with the volume of data being transported across it. As the load upon the system increases, the amount of data transferred around the system will grow. This can cause problems if there is not enough available bandwidth, such as failed data transfers, and failed writes to the data grid. Such problems should show up during stress testing.

### **Impact**

Potentially high.

Network problems can make any node unavailable, leading to any of the problems associated with [QR-10].

Network problems may also cause data transfer to be slow, or fail. This is particularly the case under high network loads.

### **Priority**

Moderate.

It is envisaged that load testing will reveal any potential network related problems.

### **Control**

- The load testing phase of the project will contain test cases that subject the network to high loads
- Create policy for minimum network requirements for new nodes.

### **Mitigation**

- Introduce better network connections at troublesome sites;
- Re-engineer the software should its network activities become a problem.

## **QR-14 Data becomes corrupt**

### **Definition**

Data or metadata on the grid may be corrupted.

### **Description**

Data on the grid may become corrupted for a number of reasons.

At present, binary data files (e.g. configurations) are stored on at least two nodes of the grid. Periodic checks are carried out to ensure that copies are exact replicas of each other. This is presently carried out by generating MD5 hashes for each copy of the file on the grid and ensuring they match. In the event that the hashes are found not to match, a “worst case” stance is taken and the data grid system is shut down completely. This leaves the data grid system unavailable until the cause of the corruption is found and integrity is restored manually, which is a non-trivial task.

Metadata is stored in an XML database (eXist, [16]) and as data files on the grid. Similar corruption issues exist for the raw XML files, though it may be easier to detect this as corrupt files are unlikely to parse as valid XML. Corruption of the XML database could also occur, leaving the metadata query system unavailable, for both “internal” grid users and the future ILDG system. Currently the database is not backed up, which means that reintegration needs to be done manually from the data files on the grid.

Incorrect data may be entered onto the grid by mistake. At present, the policy for modifying data on the grid is unclear.

### **Related Risks**

Node unavailability [QR-10].

### **Likelihood**

Moderate.

Data corruption on hard drives is rare. This situation has occurred once during the normal operation of the system over the last two years, due to a RAID drive failing. There is a possibility that scaling up of the system may make this more likely, but excepting rare cases like drive failure, this risk carries a low overall likelihood.

Similarly, the likelihood of the XML database becoming corrupt during normal operation is low.

The likelihood of data loss due to nodes retiring is estimated to be low, as a high rate of node retirement is not envisaged.

There is a moderate likelihood that malicious or unknowing user action could cause data corruption. For example, a user could run a job on one of the storage nodes that damaged the contents of the hard drive. This is however unlikely, as the user would need to have an account on that machine, and the storage nodes are dedicated to storing data and not normally used for running large jobs. There is also the risk of malicious damage by hackers, or that a storage node is hit by a virus.

There is a high likelihood that incorrect data will be placed on the grid at some stage. At present it is necessary to have an administrator remove this data and then replace it with correct data. It may be desirable in the future to grant the ability to update data using command line tools.

### **Impact**

High.

When corruption is detected write access is totally denied, which has implications for those wishing to use the system. This particularly affects users who have scheduled jobs to automatically dump data to the grid – this may cause blocking problems on other nodes wishing to write. There is a possibility that

halting write access may not stop further corruption – for example, in the case of a virus propagating from machine to machine.

**Priority**

Moderate.

**Control**

- Avoid downtime by implementing automatic consistency checking – by keeping a record of the original MD5 checksum for each file when the data is placed on the grid it is possible to tell which duplicate of a file has become corrupted – this file could then be removed and replicated. The record could be stored in the replica catalogue, in the metadata (this may require an update to the QCDML schema) or in a separate database;
- Implement a backup procedure for the XML database containing the metadata files, perhaps offering a read-only mechanism while reintegration takes place;
- Ascertain whether there is a real requirement for developing a command line tool for modifying or versioning data, and whether effort is available to implement such a tool;
- Implement a system for detecting corruption of XML database and restoring the database from the grid when it is found to be corrupt.

**Mitigation**

At present the system is shut down completely when data corruption is detected. Depending on which of the controls above are implemented, this may or may not be an appropriate course of action. In any event, keeping the user community informed of any changes to the system is a priority.

**QR-15      Change in requirements and bug reporting****Definition**

The requirements of the QCDgrid user community for the present system change.

**Description**

At present, the QCDgrid system fulfils the following broad functionality: store data and metadata in a reliable and searchable manner; allow the retrieval of the data; allow the submission of simulation or post-processing jobs to a variety of systems.

The needs of the user community may change. For example, the type of job and nature of the machines used may evolve over time. Jobs may change from straightforward simulations to post-simulation analysis jobs on smaller machines. Non-ensemble data (for example, results of post-simulation analysis) will have to be stored on the grid, along with the associated metadata (at the moment the metadata catalogue deals with QCDML documents only). As pointed out above, the grid may have to deal with larger amounts of data. Defects may be found in the software and will need to be remedied. These changing requirements will have to be reflected by the software.

At present changes to the software and bug reports are dealt with as required. A formal versioning and release system has not been adopted. There is a risk that desired changes may not be carried out, due to there not being enough effort to implement them, or because they are not tracked properly.

**Likelihood**

Moderate.

There is no system for dealing with change requests or bug reports. At present, such requests are filed via the NeSCForge site or by email and dealt with on an ad-hoc manner. Large changes to the system functionality are not expected at this point in time. However, bug reports will continue to be filed. A release plan for software developed in the future will be useful.

**Impact**

High.

- Required changes may not be carried out and bugs may not be fixed;
- Managing user expectations becomes difficult;
- The software maintenance effort is limited, and may not be used effectively.

**Priority**

Moderate.

**Control**

- Develop a release plan, incorporating mechanisms for requesting changes, a versioning scheme and proposing a release schedule (e.g. quarterly, monthly). Involve the community in deciding which changes should go into the software;
- Use the software maintenance budget efficiently to accommodate changes in requirements;
- Be aware of the fact that the requirements may change dramatically, and acknowledge that the work-plan will need to be altered in this circumstance;
- Co-operate with the user community to make sure the release plan reflects real requirements changes.

**Mitigation**

This will be mitigated by implementing the above controls. Effort has already been allocated for developing a release plan.

## 4 Conclusions

Having analysed the risks, it is clear that there are a number of high priority, high impact risks. Chief amongst these is the fact that the system has not been thoroughly stress tested in preparation for the load increasing (QR-6). We will take direct action to control this risk immediately. Other moderate priority risks include lack of an automated test infrastructure (QR-7), node unavailability (QR-10), security risks (QR-11), insufficient support for job submission (QR-12) and data corruption (QR-14). These risks will be tackled should time become available within the work plan. The rest of the risks are less serious and require either monitoring or the introduction of control policies, rather than immediate direct action.

The rest of this section describes a control plan, with the actions being drawn from the risk controls. Based on the above risk analysis we propose the following action categories and actions. We also discuss the allocation of effort from the work plan that will allow us to carry out these actions. Specific actions are numbered [A-N] where N is an integer.

### **Implement testing suite, including stress testing**

- [A-1] Gather requirements for stress testing.
- [A-2] Implement stress testing suite and carry out stress testing.
- [A-3] Test response of processes writing to the grid when nodes are unavailable.

Effort from Work Packages 1.2 (Stress Testing Requirements) and 1.3 (Stress Testing) will be used to carry out these actions.

### **Conduct security audit**

- [A-4] Conduct a security audit of the existing C code.

This audit will be conducted using the tools mentioned in QR-11. Effort from Work Package 1.5 (Maintenance and support) will be used to carry out this action.

### **Continue development of core grid system**

- [A-5] Continue development of job submission system.
- [A-6] Continue maintenance and development of data grid system.
- [A-7] Continue development of metadata browser.
- [A-8] Write a development and release plan.
- [A-9] Modify system so that when nodes are temporarily retired the data in their “inbox” is immediately replicated prior to retirement.

As further requirements emerge, effort from Work Package 1.5 (Maintenance and support) will be used to implement new features and bug fixes for the data grid and job submission systems. A release plan will be written, detailing the versioning system for the software and the release schedule.

### **Raise awareness in community**

[A-10] Update all documentation relating to installation and administration and circulate to community.

[A-11] Continue the process of switching to “pooled accounts” Globus security.

These can be carried out in the context of Work Package 1.5 (Maintenance and support).

[A-12] Ensure community are aware of means by which data on local nodes becomes corrupt (as detailed above).

This can be achieved by circulating this document.

[A-13] Encourage the community to make use of the metadata functionality by creating tools and awareness-raising (by email and announcements at meetings).

Tools that aid in the creation of metadata will be created in Work Package 2. The need to make use of the metadata functionality will be discussed regularly.

### **Future software development**

No actions have been scheduled relating to future software development; however the risks imply a number of things to be aware of during future development, during both design and implementation. These include but are not limited to the following:

- Design for scalability;
- Design with security in mind, possibly using formal security practices;
- Write sufficient tests alongside new code;
- Consider usability, in terms of installation, administration and user interfaces;
- Write software to be flexible with the choice of middleware, as this may change;
- Ensure consistency of data on the grid;
- Ensure future user documentation is clear and concise.

### **Monitoring community activities**

The QCDGrid development team should keep informed about activities in the following areas:

- Changes to the metadata schema (QCDML);
- Changes in requirements for the grid;
- Changes to the hardware and network infrastructure (in order to control any potential problems).

Both of the above may be achieved by regular attendance at video conferencing, paying attention to email lists and ad-hoc conversations with community members.

### **Monitoring third party software**

[A-14] Subscribe to relevant mailing lists regarding changes to Globus, eXist and any other middleware used in future developments to track the status of each product.

**Monitoring QCDgrid software**

[A-15] Develop a release plan that will allow new features to be added in a sensible manner, and bugs to be fixed.

## References

- [1] UKQCD Collaboration home page <http://www.ph.ed.ac.uk/ukqcd/>
- [2] The Virtual Data Toolkit, available from <http://www.cs.wisc.edu/vdt/index.html>
- [3] Setting Up Systems for QCDgrid, James Perry. Available from <http://www.epcc.ed.ac.uk/~jamesp/qcdsetup.pdf>
- [4] QCDgrid NeSCForge site, [http://forge.nesc.ac.uk/tracker/index.php?func=detail&aid=64&group\\_id=13&atid=133](http://forge.nesc.ac.uk/tracker/index.php?func=detail&aid=64&group_id=13&atid=133)
- [5] JUnit <http://www.junit.org>
- [6] Flawfinder <http://www.dwheeler.com/flawfinder/>
- [7] RATS [http://www.securesw.com/download\\_rats.htm](http://www.securesw.com/download_rats.htm)
- [8] Writing Secure Code, M. Howard & D. LeBlanc, Microsoft Press 2003
- [9] UKQCD collaboration, "GridPP Log Book", available on request from [qcdgrid-enquiries@epcc.ed.ac.uk](mailto:qcdgrid-enquiries@epcc.ed.ac.uk) (September 2004).
- [10] The Open Grid Services Infrastructure, <https://forge.gridforum.org/projects/ogsi-wg>
- [11] The Web Services Resource Framework <http://www.globus.org/wsrf/>
- [12] Unicore, available from <http://unicore.sourceforge.net/>
- [13] The Grid Application Toolkit, available via the GridLab project <http://www.gridlab.org/>
- [14] XIncid, part of the Apache Project, available from <http://xml.apache.org/xindice/>
- [15] Tamino, a product of SoftwareAG. See <http://www2.softwareag.com/corporate/products/tamino/default.asp> for details.
- [16] eXist <http://exist.sourceforge.net/>