



CONTRACT NUMBER 508830

DEISA
**DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR
SUPERCOMPUTING APPLICATIONS**

European Community Sixth Framework Programme
RESEARCH INFRASTRUCTURES
Integrated Infrastructure Initiative

JRA7 Quality Plan v4.0

Deliverable ID: DEISA-DJRA7-1.4
Due date: April 30th, 2007
Actual delivery date: May 23rd 2007
Lead contractor for this deliverable: EPCC, UK

Project start date: May 1st, 2004
Duration: 4 years

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	X

Table of Content

Table of Content.....	1
1. Introduction.....	3
1.1 Executive Summary.....	3
1.2 Document Structure.....	3
1.3 Changes between Versions 3 and 4.....	3
1.4 Changes between Versions 2 and 3.....	4
1.5 Changes between Versions 1 and 2.....	4
1.6 References and Applicable Documents	4
1.7 Document Amendment Procedure	5
1.8 List of Acronyms and Abbreviations	5
2. Activity Description	7
2.1 DEISA.....	7
2.2 JRA7 Objectives.....	7
2.3 JRA7 Organisation and Responsibilities	8
2.3.1 Activity Personnel	8
EPCC	8
FZJ	8
CINECA	8
ECMWF	9
3. Statement of the Problem.....	10
3.1 Generic Requirements.....	10
3.2 Overall Approach.....	10
4. Management Approach	12
4.1 Assumptions and Constraints.....	12
4.1.1 Dissemination	12
4.1.2 Travel.....	12
4.2 Reporting and Monitoring	12
4.2.1 Weekly Reports	12
4.2.2 Weekly activity Status Meetings	12
4.3 Risk Management.....	13
4.4 Change Control	13
5. Technical Approach.....	14
5.1 Reuse Strategy.....	14
5.2 Assumptions and Constraints.....	14
5.3 Development Environment	14
5.4 Activities, tools and products	15
5.4.1 Coding guidelines	15
5.4.2 Auto-documentation	15
5.4.3 Logging Policy	15
5.4.4 Exceptions	16
5.4.5 CVS Directory Structure	16
5.5 Build Strategy	17
6. Product Assurance	17
6.1 Assumptions and constraints.....	17
6.2 Testing Strategy	18
6.2.1 Unit Test Strategy.....	18
6.2.1.1 Layout.....	18
6.2.1.2 Guidelines for designers	18
6.2.1.3 Guidelines for implementers	18
6.2.1.4 Unit testing frequency	18
6.2.1.5 Test failures	19
6.2.1.6 Test reviews.....	19
6.2.2 System Testing.....	19

6.2.3	Manual Testing	20
6.2.4	Distributed Development – Integration Testing	20
6.2.5	Production Testing.....	20
6.3	Quality Assurance	20
6.4	Configuration Management	21
6.5	Problem Report System	21
6.6	Feature Requests	21
6.7	Support Requests.....	21
6.8	Document Review Process	22
6.9	Release Management	22
6.10	Maintenance	22

1. Introduction

1.1 *Executive Summary*

DEISA (Distributed European Infrastructure for Supercomputing Applications) is a consortium of leading national supercomputing centres that intends deploying and operating a persistent, production-quality, distributed supercomputing environment with continental scope. Joint Research Activity 7 or JRA7 as it is more commonly referred to, is a research activity within DEISA that is developing a software layer for coordinating and integrating OGSA (Open Grid Services Architecture)-based services for distributed resource management in a heterogeneous environment. This software layer is referred to as the DESHL (DEISA Services for Heterogeneous management Layer). JRA7 is also using the DESHL to provide user-level tools with access to these resource management services.

This document, JRA7 Quality Plan v4.0, is the Quality Plan for JRA7. It is the top-level document for controlling the activity and as such describes (or references) how the activity will be executed. This document provides the initial motivation for the activity and records its initial high-level requirements. The audience for this document are the participants in JRA7 and the DEISA Executive Committee.

This document will undergo major revision at the start of each new year of the activity. In addition, it can be regarded as a living document that will be updated as necessary at other points in the activity lifetime. In summary the document sets the scene for the initial activity processes that can then be adapted as the activity sees fit.

1.2 *Document Structure*

This section of the document, section 1, introduces the document and includes an executive summary, references and a list of acronyms and abbreviations. Section 2 describes the organisation of JRA7 with section 3 providing a statement of the problem that JRA7 is tackling. Section 4 describes the management approach including handling of risks, reporting and monitoring with Section 5 describing the technical approach. In particular this describes the development environment. Finally section 6 describes the procedures for quality control and assurance and includes the testing guidelines.

1.3 *Changes between Versions 3 and 4*

This is deliverable DEISA-DJRA7-1.4 from Task 1.5 'Produce PQP v4.0' in Work Package 1 of the DEISA JRA7 activity [1]. This deliverable is therefore an updated version of the JRA7 Quality Plan v3.0 as described in deliverable DEISA-DJRA7-1.3. In brief, Version 4 differs from Version 3 as follows.

- Section 1.6 updated to include newly referenced documents
- Section 4.1.2 refers to Open Grid Forum, OGF, instead of Global Grid Forum, GGF.
- Section 5.2 now also refers to "Functional scope for DESHL V4.0" and the "Final design for DESHL v4.0", both of which are now listed in section 1.6.
- Section 5.3 now also refers to the deisa-jra7-deshl@forge.nesc.ac.uk mailing list.
- Section 5.4.3 updated to refer to the latest logging policy for the DESHL.
- Section 5.4.4 updated to refer to the latest policy for exception handling in the DESHL.

1.4 Changes between Versions 2 and 3

Version 3 is an updated version of the JRA7 Quality Plan v2.0 as described in deliverable DEISA-DJRA7-1.2. In brief, Version 3, that is deliverable DEISA-DJRA7-1.3, differs from Version 2 as follows.

- Section 4.2.2 notes that activity status meetings are now held weekly. Previously these were held fortnightly.
- Section 4.3, risks are now reviewed at the weekly activity status meetings and if necessary reported in the 6 monthly progress reports. Previously progress reports were quarterly and risk reviews were fortnightly.
- Section 5.1, now only refers to the relevant installation documentation and build files for listings of software components from other sources utilised in the activity.
- Section 5.3, removed reference to the out of date DESHL v1.0 development environment document, the description of tracker usage to indicate these are publicly available. Explained that users are expected to use the trackers rather mailing lists.
- Section 5.4.5, updated to refer to components of DESHL v3.0.
- Section 5.4.5 updated to refer to DESHL v3.0 situation.
- Section 6.2.2, updated to describe the system testing environment.
- Section 6.5, updated to indicate that bug tracker is now publicly available
- Section 6.6, added information on publicly available feature request tracker.
- Section 6.7, added information on publicly available support request tracker.

1.5 Changes between Versions 1 and 2

Version 2 is an updated version of the original JRA7 Quality Plan as described in deliverable DEISA-DJRA7-1.1. In brief, Version 2 differs from Version 1 as follows.

- Complies with the DEISA document style.
- Removal of content contained in other DEISA deliverables eg. staffing budgets as listed in [1].
- Change of content to reflect the first year of DEISA ie.
 - The Technical Approach section has been updated to reflect the software development process agreed amongst JRA7 participants.
 - The Product Assurance section has been updated to describe the testing guidelines agreed amongst JRA7 participants

1.6 References and Applicable Documents

- [1] DEISA Annex I – “Description of Work”, November 5th 2003.
- [2] DEISA public web site, <http://www.deisa.org/>
- [3] UNICORE, Uniform interface to computing resources, <http://unicore.sourceforge.net/>
- [4] Functional scope for HSM v1.0, DEISA-JRA7-3.1 v1.0
- [5] Functional scope for DESHL v2.0, DEISA-JRA7-3.6 v1.0
- [6] Functional scope for DESHL v3.0, DEISA-JRA7-3.9
- [7] Functional scope for DESHL v4.0, DEISA-JRA7-3.11
- [8] Final design for HSM v1.0, DEISA-JRA7-3.4 v1.0

- [9] Final design for DESHL v2.0, DEISA-JRA7-3.7 v1.0
- [10] Final design for DESHL v3.0 (This document is not a DEISA EU deliverable but is available under Docs at [15]. The DESHL v3.0 software release can also be downloaded from [15] .)
- [11] Final design for DESHL v4.0 (This document is not a DEISA EU deliverable but is available under Docs at [15]. The DESHL v4.0 software release can also be downloaded from [15].)
- [12] Final design for DESHL v4.1 (This document is not a DEISA EU deliverable but is available under Docs at [15]. The DESHL v4.1 software release can also be downloaded from [15].)
- [13] NeSCForge, <http://forge.nesc.ac.uk/>
- [14] CVS, Concurrent Versions System, <http://www.cvshome.org/>
- [15] DEISA JRA7 NeSCForge web site, <https://forge.nesc.ac.uk/projects/deisa-jra7/>
- [16] Commons Logging, Apache Jakarta Project, <http://jakarta.apache.org/commons/logging>
- [17] XMLBeans, <http://xmlbeans.apache.org/>
- [18] Javadoc, <http://java.sun.com/j2se/javadoc/>
- [19] GForge, <http://gforge.org/>
- [20] Junit, <http://www.junit.org/index.htm>
- [21] Ant, The Apache Ant project, <http://ant.apache.org/>

1.7 Document Amendment Procedure

The procedure for amending this document is described in section 6.8 .

1.8 List of Acronyms and Abbreviations

AOB	Any Other Business
BSD	Berkeley Software Distribution
CINECA	Consorzio Interuniversitario
CVS	Concurrent Versions Systems
DEC	DEISA Executive Committee
DEISA	Distributed European Infrastructure for Supercomputing Applications
DESHL	DEISA Services for Heterogeneous management Layer
ECMWF	European Centre for Medium-range Weather Forecasting
EPCC	Edinburgh Parallel Computing Centre
FZJ	Forschungszentrum Julich
GGF	Global Grid Forum

HPC	High Performance Computing
HSM	Hierarchical Storage Management
HSM	Heterogeneous Service Management (now superseded by DESHL)
JRA1	Joint Research Activity 1 – Materials Sciences
JRA2	Joint Research Activity 2 – Cosmology
JRA3	Joint Research Activity 3 – Plasma Physics
JRA4	Joint Research Activity 4 – Life Sciences
JRA5	Joint Research Activity 5 – Industry
JRA6	Joint Research Activity 6 – Coupled Applications
JRA7	Joint Research Activity 7 – Access to Resources in Heterogeneous Environments
N1	Networking Activity 1 - Management
N2	Networking Activity 2 - Dissemination
OGF	Open Grid Forum
OGSA	Open Grid Services Architecture
PM	Project Month
PQP	Project Quality Plan
SA1	Service Activity 1 - Networking
SA2	Service Activity 2 – Data management with global file systems
SA3	Service Activity 3 – Resource Management
SA4	Service Activity 4 – Applications and User Support
SA5	Service Activity 5 - Security
UNICORE	Uniform Interface to Computing Resources [3]
XML	eXtensible Markup Language

2. Activity Description

2.1 DEISA

DEISA (Distributed European Infrastructure for Supercomputing Applications) is a consortium of leading national supercomputing centres that intends deploying and operating a persistent, production-quality, distributed supercomputing environment with continental scope [2]. DEISA is funded by the European Commission under the Sixth Framework Programme.

The DEISA infrastructure consists of two layers: a “core” layer consisting of a strongly coupled (via a dedicated network) multi-cluster of homogeneous supercomputers, and an “extended” Grid layer. The Grid layer consists of a variety of weakly coupled heterogeneous computing resources provided partly by the project partners, and partly by other organizations or infrastructures that work with the DEISA infrastructure. ([1], Section A, 5.C.8 page 219)

AIX is the operating system within the homogeneous core layer of the DEISA infrastructure. The heterogeneous extension refers to incorporation of and interoperability with Linux operating systems ([1], page 22).

The global project management of DEISA is undertaken by an activity referred to as N1. A further activity, N2, is concerned with the dissemination of knowledge and experiences resulting from the operation of the infrastructure.

The operation of the DEISA infrastructure is the responsibility of 5 Service Activities (SAs) covering networking (SA1), data management with global file systems (SA2), resource management (SA3), applications and user support (SA4), and security (SA5). Joint research activities (JRAs) will exploit the DEISA infrastructure. The first 6 of these JRAs are applications oriented and cover Materials Sciences (JRA1), Cosmology (JRA2), Plasma Physics (JRA3), Life Sciences (JRA4), Industry (JRA5) and Coupled Applications (JRA6).

The seventh joint research activity, JRA7, is titled ‘Access to Resources in Heterogeneous Environments’ and is concerned with the development of a Heterogeneous Service Management software infrastructure based on OGSA (Open Grid Services Architecture) standards ([1], Section 4.3 page 25). JRA7 will develop the advanced software service layers needed for the management of distributed, loosely coupled, heterogeneous resources and may develop or enhance required software services to manage the DEISA super-cluster, based on requirements from the Service Activities. This basic middleware development is needed to enlarge the capabilities of the infrastructure and to cope with fast technology evolutions. This activity will help provide efficient resource management interfaces for the second and third generations of the DEISA middleware stack ([1], Section A, 5.C.8, page 219).

2.2 JRA7 Objectives

DEISA JRA7 will develop a means for coordinating and integrating OGSA (Open Grid Services Architecture)-based services for distributed resource management in a heterogeneous environment, the heterogeneous service management layer or DESHL, and to use this to integrate a variety of existing user-level tools to provide the necessary high-level services in, for example:

- Authentication, authorisation and accounting;
- Job preparation, submission and monitoring;
- Data movement for job input and output;
- Other areas to be determined by DEISA user requirements.

[1], see Objectives, page 209)

Please note that the acronym, HSM, is commonly used to refer to Hierarchical Storage Management. To avoid possible confusion within this document and future JRA7 documents, the acronym DESHL is used to refer to the heterogeneous service management layer rather than the original acronym, HSM. DESHL is derived from the term '**DEISA Services for the Heterogeneous management Layer**'. This term more accurately describes the purposes of this layer in the DEISA infrastructure. The acronym HSM is as far as is possible only used in the titles of deliverables previously named in the DEISA Description of Work [1].

2.3 JRA7 Organisation and Responsibilities

There are four participants in JRA7. These are EPCC, FZJ, CINECA and ECMWF. EPCC is the activity leader for JRA7 and is responsible for producing the main deliverables, collecting the contributions of the participants, keeping deliverable deadlines and for producing the final report for the activity (section C, page 30, [1]).

A Project Manager at EPCC is the Activity Leader for JRA7. The JRA7 Quality Plan (this document) governs overall activity and quality management. This Quality Plan was initially produced during PM (Project Month) 1 and is updated on an annual basis. The Quality Plan covers all aspects of activity standards, activity tracking, monitoring and reporting, risk management, quality control and quality assurance. EPCC has responsibility for producing and applying the Quality Plan, and all JRA7 participants agreed to implement the Quality Plan at their own sites for JRA7 work. (see section 2.1, pages 210-211, [1])

Within the broader work package structure described in [1], individual detailed tasks are identified by the JRA7 consortium, and allocated to the most appropriate members. These tasks are recorded in a detailed work plan as appropriate and task leaders for each are nominated (see section 2.1, pages 210-211, [1]). Task leaders have a key role in the tracking of and reporting on activities.

2.3.1 Activity Personnel

EPCC

The EPCC team initially consisted of 3 staff: a project manager (the JRA7 Activity Leader), a consultant and a technical reviewer. The Activity Leader manages JRA7 as a whole and leads those tasks undertaken at EPCC. The consultant undertakes those technical tasks assigned to EPCC with the technical reviewer providing assistance in the quality control and assurance of those tasks.

In the initial JRA7 tasks, EPCC primarily participate in the assessment of user-level tools and provided some input on the evaluation of standards. Following this, EPCC has focused and will continue to focus on the development of the DESHL and its integration with user-level tools.

FZJ

FZJ have assigned staff to JRA7 tasks as necessary and will continue to do so. In the initial tasks, FZJ primarily participated in the evaluation of standards and provided some input on the assessment of user-level tools. Since then FZJ have participated in the development of the DESHL and its integration with user-level tools.

CINECA

CINECA have assigned staff to JRA7 tasks as necessary and will continue to do so. CINECA provide feedback from the DEISA Resource Management Service Activity

(SA 3). In the initial tasks, CINECA collected user requirements from the partners in the DEISA consortium.

ECMWF

ECMWF have assigned staff to JRA7 tasks as necessary and will continue to do so. In the initial tasks, ECMWF primarily participated in the assessment of user-level tools. Since then ECMWF have participated in the development of the DESHL and its integration with user-level tools.

3. Statement of the Problem

3.1 *Generic Requirements*

JRA7 is defining suitable heterogeneous resource management interfaces based on existing and emerging standards from the web services and Grid communities. JRA7 is producing an implementation of these interfaces – the DEISA Services for heterogeneous Management Layer (DESHL) - for the DEISA heterogeneous Grid. These services will act as proxies for the great variety of resources deployed in a heterogeneous Grid. Tools that provide the required user-level functionality will be identified and ported to this new interface. ([1], section 1.1, page 204)

Management of resources in a heterogeneous environment requires a number of key user-level tools and services to operate transparently on top of a disparate set of underlying technologies. Integrating this functionality from a range of tools is the main challenge for JRA7. Users and administrators of the DEISA infrastructure need as minimum, services covering:

- Authentication, authorisation and accounting (cf. SA5 – Security)
- Job preparation, submission and monitoring (cf. SA3 – Resource Management)
- Data movement for job input and output (cf. SA2 – Data)
- Monitoring and Management of resources

JRA7 is addressing each of these core service areas – working closely with the appropriate service activities – focusing on integrating functionality to provide the user with a transparent view of the heterogeneous Grid. ([1], section 1.1, page 203)

Standards are emerging from the world of Grid and web services for service description, workflow coordination and choreography. JRA7 has assessed and is continuing to assess these standards to determine if they can be adopted to provide the glue necessary to manage, via service interfaces, the resources of the DEISA Grid ([1], page 203).

3.2 *Overall Approach*

An iterative development of the necessary software for the DESHL is being employed, with major releases in every year of the activity. This is to allow the exploitation of emerging standards and tools. The intention is to identify at each stage key user-level functionality that can be delivered by suitable tools, mapping this onto a suitable functional scope for the DESHL so delivering an incremental improvement in the capabilities that the DESHL can support ([1], page 205).

Moreover it is important to recognise that the JRA7 objectives have a large scope. It is therefore imperative to restrict this focus to achievable proportions and to exploit deliverables from other grid projects as far as possible.

An iteration of the development consists of the following phases.

1. **Planning:** An initial development plan outlining the major tasks and their deliverables will be produced. This plan will be refined and updated as the iteration progresses with detailed task breakdowns added as the extent of the major tasks becomes clearer.
2. **Evaluation & Assimilation:** Existing user-level resource management tools and standards will be examined and evaluated for their applicability to the DEISA extended Grid infrastructure. Where applicable, this phase will also assess if deliverables from other grid projects can be exploited. Potential end-users will be interviewed and user authorities identified. This is effectively a

requirements collection and analysis process to determine the functional scope and priorities for the subsequent development phases. This functional scope will be used to determine the system test cases for this development iteration.

- 3. Development:** Prototyping and early design to meet the desired functional scope will be undertaken. Test plans and a specification of the testing infrastructure will be produced. A cycle of design, implement, unit test, system test and documentation tasks will iterate towards the desired functional scope. The DESHL will be released to the DEISA community.
- 4. Tool Integration:** Previously identified user-level tools will be ported to the DESHL and released to the DEISA community.

Development work will complete by Project Month 48.

4. Management Approach

4.1 Assumptions and Constraints

This section lists those points that have been noted as providing constraints on the management approach to the activity.

4.1.1 Dissemination

The overall DEISA dissemination activity, N2, will request contributions from JRA7 for DEISA publications or events. JRA7 is not expected to participate in any other specific dissemination task, however, all JRA7 participants are encouraged to produce publications and make presentations at events outside DEISA. However it will be important to ensure that the production of such extraneous publications and presentations has a minimal impact on the overall priorities for JRA7.

The DEISA Executive Committee must be informed about any intended publications or presentations. Copies of publications and presentations must be lodged with the N2 activity leader.

4.1.2 Travel

FZJ and EPCC will work in the relevant Working and Research Groups of the Open Grid Forum (OGF – formerly Global Grid Form, GGF)). FZJ will assume leadership roles as co-chair and author and if possible in the management of OGF. This implies continued participation for one FZJ and one EPCC staff at three meetings per year, at least two of which will be outside Europe, in the US or Asia-Pacific region ([1], page 213, section 2.2, note 2).

JRA7 will be integrated in the same management structure as the SA and as such, will get its cost for attendance at the DEISA management meeting covered under activity N1 (management) ([1], page 213, section 2.2, note 3).

4.2 Reporting and Monitoring

4.2.1 Weekly Reports

Once a week, the Activity Leader will contact the representative of each participant currently undertaking tasks on the activity. The Activity Leader and the representative will discuss progress on the currently active task.

4.2.2 Weekly activity Status Meetings

Meetings of the full activity team are held at least weekly at a regular time. The Activity Leader chairs this meeting. The meeting is primarily held via teleconference though on occasions Access Grid may be used.

The purpose of this meeting is to allow the Activity Leader to monitor progress and to provide a forum to discuss issues as they arise. An action list and issues log is maintained and reviewed at this meeting.

The agenda of this meeting is as follows:

1. Review of ongoing actions
2. Status reports from each activity participant
3. Review of current work plan
4. Review of issues and risks; logging of new issues
5. AOB

4.3 Risk Management

This section describes the strategy to be adopted for managing risks that threaten a successful outcome to the activity.

All risks, and issues which develop from risks, will be recorded in the minutes of the weekly activity status meeting under the risk log. All risks that arise before or during the activity will be noted in this log, as will all risks that become issues. The risk and issue log is a live document that will be reviewed at each weekly activity status meeting and will be updated as required.

Risks and issues are escalated if necessary to the DEC and if so are reported in the 6 month progress reports.

4.4 Change Control

A process for managing changes in requirements is an essential part of the software management process. We adopt a fairly simple, lightweight approach for decisions internal to JRA7.

1. The Activity Change Board consists of a representative from each of the 4 participants in JRA7 for it to be quorate. The representative must have sufficient authority to approve or disprove a change request.
2. Change requests are made in writing (email is fine) to the change board, stating the following:
 - reason for change;
 - probable impact of change on software, schedule, budget etc.;
 - probable impact of *not* making the change;
3. Each change request is discussed by the change board (again, email is fine; the outcomes of a phone call must be documented and disseminated after the call, for confirmation);
4. The decisions of the change board are binding.
5. All change requests, plus their outcomes, must be logged.

5. Technical Approach

5.1 Reuse Strategy

This section outlines if software components from other sources are to be utilised in the activity.

JRA7 software development will, where licensing conditions permit, use existing software components from other sources.

A full list of the software components necessary for using the JRA7 developed software will be included in the relevant installation documentation and build files.

5.2 Assumptions and Constraints

This section outlines any technical assumptions or constraints for the activity. These are primarily listed in the JRA7 scope documents ([4], [5], [6] and [7]) with any further additions listed in the JRA7 design documents ([8], [9], [10],[11] and [12]).

In addition to these constraints there is a further obvious constraint that JRA7 is undertaking distributed software development. To deal with this, each JRA7 participant is responsible for a particular set of components during a development cycle. This results in a dependency on key interfaces between JRA7 participants. Agreement on these key interfaces between JRA7 participants is important. The change control procedure described in section 4.4 is therefore employed to deal with changes to these key interfaces.

5.3 Development Environment

This section describes the development environment for the activity. As previously mentioned JRA7 is undertaking distributed software development. It has therefore been important to use suitable collaborative software development tools to enable easy communications and interactions between the JRA7 participants. The principal tool employed to assist JRA7's collaborative development is NeSCForge [13]. This is a GForge system [19] hosted at the UK's National e-Science Centre. A DEISA JRA7 project has been set up on NeSCForge.

In particular, NeSCForge is used to provide the following facilities for JRA7 software development.

- A DEISA JRA7 web site [15] to enable distribution of software and allow developer access to the JRA7 specific information.
- Revision control management (CVS [14]) repository for all JRA7 developed code and documentation
- Bug and issue trackers. There are 4 public trackers for bugs, support requests, patches and feature requests.
- Mailing lists. The `deisa-jra7-developers@forge.nesc.ac.uk` and `deisa-jra7-deshl` mailing lists are both hosted via the JRA7 NeSCForge. The first list is used for JRA7 development specific discussions with the second available for user discussions. Only the former is heavily used at present. The `jra7@deisa.org` mailing list hosted by IDRIS is used for more general JRA7 issues. To ensure that user requests and issues are filed and tracked, DESHL users are encouraged to use the JRA7 NeSCForge public trackers for support requests and issues.

The NeSCForge facility for JRA7 is managed by EPCC and is available at [15].

5.4 Activities, tools and products

This section describes any other activities, tools or products to be used in the activity eg coding conventions, auto-documentation tools, and so on.

5.4.1 Coding guidelines

The Sun coding guidelines as listed at <http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html> will be followed.

In particular, all JRA7 developed code files should contain a header similar to that shown in the example in Figure 1.

```

/*
 * Title:          DESHLCommand.java
 * Description:    Represents class that is able to process a command for the CLT.
 * Author:        Thomas Seed
 * Created:       21-Feb-05
 * CVS:          $Id: DESHLCommand.java,v 1.2 2005/04/18 11:16:57 tseed Exp $
 *
 * Copyright (c) 2005 University of Edinburgh
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * * Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 * * Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in
 * the documentation and/or other materials provided with the
 * distribution.
 * * Neither the name of the University of Edinburgh nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
 * THE POSSIBILITY OF SUCH DAMAGE.
 */

```

Figure 1: Example source code header

5.4.2 Auto-documentation

All JRA7 developed classes and public entities must be documented with javadoc [18] comments. This includes test classes.

Any detailed design will be the generated javadoc. Implementation classes must be clearly documented using javadoc. Each implementation of a public interface must have a link to a test class. Public interfaces and their test classes must be clearly documented using javadoc.

5.4.3 Logging Policy

Section 8 of the 'Final design for DESHL v4.1' ([12]) outlines the current logging policy for the DESHL. In particular, to avoid cluttering logs, more careful consideration has been taken on whether to log an exception when it is caught.

5.4.4 Exceptions

Section 8 of the 'Final design for DESHL v4.0' ([12]) outlines the current policy for exception handling in the DESHL. This has been formulated to help deal with the previous reports that DESHL error reporting was developer-centric rather than user centric. Also of note is that where appropriate SAGA –based exception handling is implemented.

5.4.5 CVS Directory Structure

This section covers the CVS policy for each component that JRA7 develops. Each component is considered as a sub-project and is placed under its own directory tree that is managed by CVS. The NeSCForge CVS repository is used for these components and the parent directory is called `dev`. Each component is given a suitable named directory under `dev`, for example the Client Interface Library component is called `desh-client-intf`. It is expected that components will follow the following structure:

- source
 - separating server/client/common,
 - Java/scripts,
- tests
 - unit tests – mirror source directory
 - system tests
 - manual tests (if appropriate)
- docs

More specifically this can be represented as shown in Figure 2.

```
+-- <component>
  +- src/
    | +- client/
    | | +- java/
    | | | +- org/
    | | +- scripts/
    | +- common/
    | +- server/
  +- test/
    | +- unit/
    | | +- client/
    | | | +- java/
    | +- system/
    | +- manual/
  +- doc/
```

Figure 2: NeSCForge CVS repository directory tree

Elements of each component are checked in by the developer regularly. The repository is considered to be in a permanent state of development. When a component is ready for release it is tagged. Only a tagged version of a component should be considered as usable externally. Tagged versions will also be available from the project on NeSCForge [15]. The CVS is generally for internal development.

5.5 *Build Strategy*

This section describes the build environment for the activity. Build and deployment uses Ant [21]. Each component has its own build file, `build.xml`, an Ant file that captures the build process for the component. This process includes:

- initialisation – generating time stamps, creating directories
- source code generation – some code may be auto-generated from schema files
- source code compilation – generate executable code
- testing – run the component tests
- documentation – generate documentation from the source code
- release bundle – package the executable code and the documentation for release

This build file is included at the top level of the component's CVS directory. Components are likely to have external dependencies, as listed in the release documentation. These are captured in the build configuration file, `build.properties`. As this file is likely to be different for each developer, reflecting their own environment, a template for it is included in the CVS. It may be possible to provide a super build file that can manage building all the components using each component's build file.

6. Product Assurance

This section describes (or references) the processes to be used for quality control, assurance and maintenance. This section does **NOT** refer to adherence to ISO9001 or similar quality standards.

6.1 *Assumptions and constraints*

This section outlines any assumptions or constraints associated with the product assurance and maintenance activities. Typically these would require that a quality definition be agreed upon at the start of the activity. This should be a set of measurable and quantifiable criteria that bear in mind the ultimate usage of the products. The quality definition should also be updated as part of any quality assurance exercises.

The JRA7 quality objective is to monitor adherence to the JRA7 Quality Plan throughout the project lifecycle. The JRA7 Quality Plan is designed to provide for the assurance of quality, according to the main JRA7 characteristics.

The JRA7 characteristics are:

- Cooperation of partners from European HPC centres,
- Distributed development of software (4 participants),
- Software heterogeneity i.e. AIX and Linux.

JRA7 must address the following quality factors

- Conformity: conformance to requirements
- Efficiency: the software must use resources in the best way
- Reliability: the software must run without anomaly
- Flexibility: aptitude of the software to evolve with appearance of new requirements
- Maintainability: easy to debug and correct
- Reusability: able to be used as a component in other software
- Testability: complete test suite for non regression testing
- Documented: the deliverables must be well documented

- Security: JRA7 developed software must be secure.

6.2 Testing Strategy

This section describes a possible testing strategy to be employed by the activity.

As far as possible all tests will be automatic rather than manual.

6.2.1 Unit Test Strategy

These guidelines define a possible unit testing policy for the activity. It should be noted that these are Java/Junit [20] specific.

6.2.1.1 Layout

- Unit test classes should be placed close to packages they test.
- Keeping test classes in a separate directory allows testing at the package, protected and public level.
- A suite should be available to invoke all unit tests under a given package.
- Wrapper scripts to execute suites of unit tests should be available (most often you may want to execute a portion of unit tests, which will be built as suites, testing important functionality of a package. But, the suite that executes all unit tests should be run regularly).
- JUnit should be used for Java classes.
- JUnit tests should exist for all the public operations of the Client Library interfaces.
- JUnit tests should exist for all the implementing operations of the DESHL operations.

6.2.1.2 Guidelines for designers

- Design a test sub-package for each package. For example the package `org.dai.gds` will have an associated test package `org.dai.gds.tests` where all unit test classes are to be placed.
- A unit test class should be designed for each of the class in a package.
- For each class in the design list the important functionalities and then write down tests for them. This will ensure that all functionality that is being considered has a test.
- Make sure tests exist for at least all public methods.

6.2.1.3 Guidelines for implementers

- Implement all unit test classes following the layout described above. If you are implementing any additional tests then either update the design with these new additions or pass them to the designer.
- For each class, list boundary conditions and write down tests for boundary conditions.
- For each class, list unexpected things that could happen to it and then write down tests for them.

Run unit tests regularly as described below.

6.2.1.4 Unit testing frequency

All unit tests (suites) should be run regularly. How often depends on the time it takes to run them. If it takes too long to run them all, run them at the end of the day. Multiple test suites in this situation are useful. That is, it may be necessary to write suites that test important functionality and one that tests all units in a package. Then a suite that tests an important functionality can be run when

ever the software is compiled, whereas the suite that tests all units can be run at the end of the day.

Note, the (planned) automatic nightly build process will run all unit tests and then inform developers if their code fails any unit tests. Therefore, ensure that your codes pass all unit tests prior to checking them into the code repository.

6.2.1.5 Test failures

We aim for 100% of all unit tests to pass. If the number of failures within a package is small then problems can be solved as soon as they occur. However, if the number of test failures is large (e.g., change in a class interface breaking a large number of other classes that uses it) then the test failures should be logged into a prioritised list (use the NeSCForge JRA7 bug tracker for this) and then solved accordingly. This will be helpful in a last minute code change that breaks an unexpected number of units.

If a test finds a defect that it was not designed to find then another test should be written to test for the that defect.

6.2.1.6 Test reviews

Perform regular test reviews (these may be used instead of code review). Make sure the unit test covers *important* functionalities (at least all public methods) thoroughly. **100% tests passes are not useful if unit tests do not test their classes for important expected or unexpected behaviours.** Test reviews will expose test classes to others and they may be able to help you identify missing tests in a test class (unit test). Note, testing software with inadequate tests will lead to a false sense of security; therefore allow time to review your tests.

6.2.2 System Testing

These guidelines define the general system testing policy for the activity.

- Ideally each requirement/use case in the scope deliverables (eg. [4]) should be shown to be met by a System test.
- Where possible, tests should be automatic, however to ensure adequate test coverage it may be necessary to write a manual test (see section 6.2.3).

More specifically within JRA7 for the DESHL, a system test framework based around JUnit [20] and ant[21] is being used. These are both widely used tools for Java-based development. JUnit is mostly used for unit testing. Ant is used to structure the complete development process - configuration, building, testing, deployment.

The JUnit library is used to create tests which, rather than test in a unit fashion, test at the system level. In order to replicate the actions of a user the DESHL is called as a separate process as it would be on the command line and not via any of its API. A system test class corresponds to a use case for the DESHL, for example the 'Submit a job use case' has a system test TestSubmitJobUsecase. Each test method of that class corresponds to bullets of that use case that should be verified. Taking the 'Submit a job use case' one of the bullets asserts that a user must have valid authentication, so one of the test methods checks, testNoAuthentication, verifies that if the user has no authentication then DESHL fails. Test classes can be found under the dev/ct/test/system/java directory in the JRA7 NeSCForge CVS repository.

Each particular test method may need to be run against more than one set of possible user input. While it is not possible to test against every input, the more cases that can be tested the better. An xml based configuration mechanism has been used to allow multiple inputs to be tested. This configuration mechanism utilises the XmlBeans[17] library to allow straightforward programmatic access to the test

configuration. The schema file, config.xsd, defines the configuration data for the tests (under dev/clt/schema/test/system). This structures the configuration in the following hierarchical way:

- at the top level DESHL configuration data, and each use case
 - for each use case, use case wide configuration, then each test method
 - for each test method, all the input sets that are to be tested.

A helper class for the system tests org.deisa.jra7.deshl.test.Config uses properties to find the actual xml instance document to load up the test configuration from. This document is validated against the schema. When the system tests are run each test method access the input and configuration data corresponding to the use case it belongs to through an XmlBeans representation of that xml data.

Ant is used to generate the XmlBeans from the schema, compile the test classes and run the tests using the targets genXmlBeans, compileSysTests, and sysTest respectively (from build.xml in the clt project). The sysTest target runs all the tests by default but can also run tests individually. Output from the tests is an xml formatted report - this lends itself to be marked up for display purposes or extraction of data for further analysis.

6.2.3 *Manual Testing*

In some cases an automated test may be impractical – for example checking that the UNICORE job id that the user receives is correct. Here are the guidelines for producing a manual test. These are applicable to both system and unit tests.

- A textual description of the test should appear in the appropriate test directory.
- The test should be run before a release.
- Any resulting bugs should be logged and tracked as for automatic tests.

6.2.4 *Distributed Development – Integration Testing*

As previously mentioned software development in JRA7 is distributed across the JRA7 participants. Each JRA7 participant in a particular development cycle will develop locally with their own build and unit tests. Integration tests are therefore required to ensure successful integration of the software from the various participants. Initially integration tests will be carried out against a local integration of JRA7 software components before being run between against a DESHL integration between JRA7 sites. There will be a set of integration tests for each DESHL operation.

6.2.5 *Production Testing*

In due course, production tests should be run on the DEISA environment's test and production infrastructure. Ideally the tests will be run at as many of the sites as possible. These tests will comprise the existing automatic and manual system tests.

6.3 **Quality Assurance**

The annual update of the Quality Plan (this document) is the principal means by which the activity ensures that existing practices such as testing procedures are evaluated and updated as necessary.

6.4 Configuration Management

The overall product build strategy for core functionality is informed by the following guidelines.

1. Developers start to implement tests and code based on the design.
2. Testing occurs as per design, with strategic systems tests occurring at the end of stages of core development. Unit testing carried out on an as required basis to ensure regression does not occur.
3. When the product has reached its design goals, it can be considered an "alpha" version. This is now the "current release version".
4. At this stage, it is sensible to tag the software and make a separate bug-fix stream. Any problems found while testing the alpha version are fixed in the bug-fix stream to avoid introducing new defects into the alpha stream.
5. Bug-fix streams are merged back into the main branch only after rigorous testing.
6. When the software is sufficiently stable, it will be upgraded to beta tag, and finally "1.0".
7. Bug-fix streams will be branched off the main stream as required to enable patching.
8. New functionality is usually better added to a product by branching, then merging later after development is complete. Again, this enables tracking of regression and accountability.
9. Bespoke additions and specialisations should have a founding from a main branch. This enables incremental upgrades to be added to a new branch from a bug-fix stream that originates from a main branch.

6.5 Problem Report System

The mechanism for filing and tracking problems such as bugs is described in this section.

The publicly available NeSCForge bug tracker is used for filing and tracking bugs. This can be found at <http://forge.nesc.ac.uk/projects/deisa-jra7/>.

Below are the guidelines for producing an entry in this.

- Provide (a file link, upload or description of) a test (automated or manual) that reproduces the problem where possible.
- Bugzilla provides some bug writing guidelines: <http://www.mozilla.org/quality/bug-writing-guidelines.html>
- Be as specific as possible.
- Provide a concise but specific summary.

DESHL users are encouraged to submit problems via this mechanism.

6.6 Feature Requests

The mechanism for filing and tracking feature requests is described in this section.

The publicly available NeSCForge feature request tracker is used for filing and tracking feature requests. This can be found at <http://forge.nesc.ac.uk/projects/deisa-jra7/>. All DESHL users are encouraged to use this.

6.7 Support Requests

The mechanism for filing and tracking support requests is described in this section.

The publicly available NeSCForge support request tracker is used for filing and tracking support requests. This can be found at <http://forge.nesc.ac.uk/projects/deisa-jra7/>. All DESHL users are encouraged to use this.

6.8 Document Review Process

All documents will undergo the following review process before signoff:

- document author distributes first draft and notifies the JRA7 participants on the Approval List;
- approvers read the document and comment on it, either by:
 - making direct edits *with change tracking turned on* and sending a copy back to the author;
 - calling a review meeting to discuss changes;
- changes introduced via change tracking or at discussions are accepted or otherwise by the document author;
- the above process may iterate several times as required;
- a signoff meeting is held where the release candidate draft is formally accepted by those JRA7 participants on the Approval List;
- this release candidate draft of the document is then distributed to the DEISA Project Director and the reviewer assigned by the DEISA Executive Committee;
- the approvers read the document and comment on it, either by:
 - making direct edits *with change tracking turned on* and sending a copy back to the author;
 - calling a review meeting to discuss changes;
- changes introduced via change tracking or at discussions are accepted or otherwise by the document author and the JRA7 participants;
- the above process may iterate several times as required;
- a final signoff meeting is held where the final draft is formally accepted by all those on the Approval List;
- accepted documents receive version number 1.0.

6.9 Release Management

This section describes how the release of various activity software deliverables will be dealt with.

Deliverables as listed in the DEISA Description of Work [1] will be released according to the procedures laid down by the DEISA Executive Committee. However it should be noted that JRA7 software releases, such as the DESHL, are made available via JRA7's public web pages as hosted by NeSCForge [15]. It is expected however that as JRA7 developed software becomes accepted for wider release within DEISA then it will be made available through the standard DEISA mechanisms.

6.10 Maintenance

This section describes how on-going maintenance beyond the activity end date will be handled.

Given that JRA7 developed software is open source under the modified BSD license, it is possible for maintenance beyond the end of the activity to be left to the open source community.