



JISC Project Quality Plan Template

This document defines the quality expectations the project must achieve and how they will be met.

1. Quality Expectations

The JISC programme manager completes this section defining the standards and level of quality expected to be achieved by the project.

The project will deliver the eLearning Tool(s) as specified in their proposal and refined in the JISC project plan document in line with following standards/guidelines:

- JISC (draft) Open Source Policy May 2004
- JISC (draft) Software Quality Assurance August 2004
- JISC Project Management Guidelines December 2003
- Release versions of development and final code are to placed with <http://sourceforge.net/>
- CETIS project page be maintained to communicate development progress and mapping of software to the ELF (eLearning Framework). <http://www.cetis.ac.uk/>
- Software should meet the high level functional specification as specified in the project plan.
- Software should be robust, maintainable and extendable (see JISC (draft) Software Quality Assurance August 2004).

Tolerances

- Cost – project must be completed within agreed grant.
- Time – project must be completed by 31st March 2005.
- Scope – given the short time scale of the project the scope of the deliverable (i.e. eLearning Tool(s)) may be narrowed to ensure completion on time and to budget. Any changes to scope must be agreed with the programme manager and documented via the change control procedure.
- Quality – project must adhere to the standards as defined for open standards, open source and software quality

2. Acceptance Criteria

For each of the main deliverables of the project criteria for its acceptance / completion are defined.

Successful completion of an external evaluation of the projects software outputs and development process.

3. Quality Responsibilities

List of who is responsible for monitoring and ensuring quality for different aspects of the project?

Technical Management – Will Woods, Senior Comms and IT Support Officer, IET (OU)
Pedagogy – Denise Whitelock, Senior Lecturer, IET (OU)
Project Documentation – Denise Whitelock, Senior Lecturer, IET (OU)
System Documentation – Alex Little, Lead Programmer, IET (OU)
Program issue logs, prototype versions – Alex Little, Lead programmer, IET (OU)
User assessment, usability testing and QA – Jan Rae, IET (OU)

4. Standards and Technologies

Referenced list of standards and technologies to be used by this project.

W3C standards compliance
SENDA compliance

Technologies (provisionally)

Code – Java, JSP – Java Developer Suite (Eclipse)
Operating System - Windows OS
Web Server – Apache Tomcat
Database – MySQL or MSSQL

The programming methodology will be based upon Rapid Application Development using a user centred design (UCD) approach to developing the application to meet the requirements of the end users and stakeholders. This will be achieved through prototyping and through gathering feedback from expert users and stakeholders through regular meetings during the project lifecycle.

We expect to employ a user assurance person to manage the testing process and to ensure that any significant changes are flagged up at an early stage. The stakeholders will be regularly updated on progress and their feedback will be used to ensure that the project is on target and that any changes which are required are identified at an early stage so that they can be integrated effectively into the development.

The programming development will be accompanied by verbose logging of issues and stages in the prototyping using an online “Blog” which will be available throughout the project development.

The OpenMentor code will be made available through Sourceforge and will be subject to a standard (OU) GPL licensing agreement. We will be using a version control system to ensure the integrity of the code during the development stages.

The code will be quality checked and properly documented. There will be thorough testing and robustness checks conducted at all stages to ensure that the software is stable. Usability and quality assurance will be guaranteed by the use of a “user assurance” person who will conduct testing on all aspects of the development to ensure that it meets the requirements of the user community.

UML use case and system architecture diagrams will be developed to ensure that the code is meeting correct software specifications as specified in the project plan.

5. Quality Control and Audit Processes

Description of the process to be used to control project quality and enable auditing.

At an initial workshop two processes will be begun. First, a detailed project specification will be constructed with milestones defined, and an analysis of the risks undertaken. As part of this process, any necessary additional stakeholders will be identified and their requirements will be gathered and included. Secondly, the detailed design process will begin; the software components will be identified and subsequently their interfaces defined: these will then be implemented and maintained using a version control system, which will allow parallel development among the partners, and will ensure that all code is versioned and properly archived on a daily basis.

All tasks will be tracked on a weekly basis; regular meetings will be held between the project partners to ensure that the project is following the agreed milestones and development path.. Interface design

will follow a task-centred approach (Lewis and Rieman, 1994) to ensure effective participation of stakeholders in the design process within a framework of well-defined work package deadlines

There will be quality checking at formal meetings at each significant milestone in the project lifecycle. Project milestones will fall into a pattern as follows (Note some details may change following the initial workshop mentioned earlier).

M1 – “Requirement specification and detailed development plan” – 31st October 2004 (QA1)

Provide user feedback and use case scenarios – 31st October 2004 (QA2)

Receive support and feedback from JISC in development process (QA3)

M2 – Prototype 0.1 “first Open Source version based on existing eMentor system” – 30th November 2004

T1 – Create basic ruleset to be used by OpenMentor system (QA4)

T2 – Provide specification for uploading files (QA5)

T3 – Provide database schema (QA6)

T4 – Create simple testing environment for assessment uploading (QA7)

T5 - Provide specification for parsing and analysing files (QA8)

T6 – Provide proof of concept parsing and analysis engine (standard file types) (QA9)

T7 - Provide specification for results engine (QA10)

T8 – Provide proof of concept results engine and test interfaces (QA11)

T9 - Receive support and feedback from JISC in development process (QA12)

M3 – Prototype 0.5 “developing specific OpenMentor code based on user requirements” – 31st January 2005

T1 – Provide feedback from stakeholders on prototype 0.1 (QA13)

T2 – Create assessment uploading module that can interact with parsing and analysis modules (QA14)

T3 – Create results engine which extracts valid data from database (QA15)

T4 – Extend system to deal with new rulesets based on pedagogic evaluation (QA16)

T5 - Create UI (QA17)

T6 – Perform stage 1 user assurance testing (QA18)

T7 – perform software quality checking using expert users (QA19)

T8 - bug fix and testing (QA20)

T9 - Receive support and feedback from JISC in development process (QA21)

M4 – Prototype 0.9 “Pre-release OpenMentor system” – 28th February 2005

T1 – Build extensions to system based on feedback of prototype 0.5 (QA22)

T3 - Develop UI based on output of stage 1 usability and accessibility testing (QA23)

T4 – Perform stage 2 user assurance testing (QA24)

T5 – Refine uploading, extraction, analysis and results generation modules (QA25)

T6 – perform software quality checking using expert users (QA26)

T7 – Prepare and check code documentation (QA27)

T8 – Create and check user guides and installation requirement documentation (QA28)

T9 - final pre-operational testing and bug checking (QA29)

T10 - Receive support and feedback from JISC in development process (QA30)

M5 – OpenMentor version 1.0 – 21st March 2005

T1 – Build extensions to system based on feedback from stakeholders (QA31)

T2 – Create final rulesets to be used in release version (QA32)

T3 – Make tweaks to UI based on output of stage 2 usability and accessibility testing (QA33)

T4 – Test and check all changes. Perform robustness and user assurance testing (QA34)

T5 - Receive support and feedback from JISC in development process (QA35)

Release code and documentation – 31st March 2004 (QA36)

As you can see there are various QA checks listed at each stage of the development, these will be gateways which will be checked at weekly project management meetings. If any QA gateway should be missed then the project will identify the issues and resolve these as they occur, subsequent QA checks will be made to ensure that the issues have been resolved and dependencies will be identified to ensure that development continues to run smoothly.

6. Change Control and Configuration Management Processes

Description of the process to be used to manage change and configuration management.

We will be employing a version control system, such as CVS or Microsoft Visual Source Safe to ensure that the code integrity is maintained. We have a staging server (development system) behind a firewall which we will use to deploy and test all OpenMentor prototype environments before being available for end-user or expert testing. Significant prototype changes will be made available through sourceforge <http://www.sourceforge.net>, all code changes will be documented using a blogging system and there is a shared area on the Open University Knowledge Network system where developers will share information on incremental changes and bug fixes and where all project documentation will reside. For information on the Knowledge Network visit <http://www.open.ac.uk/kn> .

We will have all code and services backed up using Legato Enterprise backup software on a nightly basis. We will also be able to recover data using the development server should the main server fail. We have standby servers available to use should a hardware failure occur. All servers are covered by an IBM service agreement and maintained in an air conditioned server room with UPS and generator backup in case of power failure.

7. Quality Tools

List any tools to be used to help ensure quality.

Tool	Description	Specific Uses
Enterprise UML	UML creation environment	Use case diagrams System specification diagrams
Knowledge Network	Knowledge Sharing system	Project documentation repository Code commenting and discussion area
Blogger	Blogging system	programming blogs
Sorceforge	Open Source code repository	Disseminate versions of OpenMentor
Legato	Backup System	Disaster recovery
VSS/CVS	Version Control system	Ensure code integrity Rollback if issues arise
IET User Testing labs (and software)	usability/accessibility testing	User assurance, system QA
BOBBY/JAWS	SENDA and website testing	Quality assurance of interface SENDA compliance W3C compliance
JUnit or similar (TBA)	Java, JSP testing environment	

