

6 May 2005

# Software Quality Evaluator Final Report

*A report on an evaluation and testing process  
commissioned by JISC*

**Authors:**

**Suzi Wells**

**Futurate Ltd**

Web: <http://www.therightplace.net/sqe/>

**William Pellett**

**Epic plc**

Email: [sqe@clax.co.uk](mailto:sqe@clax.co.uk)

**Gill Osguthorpe**

**Futurate Ltd**

**John Harris**

**Epic plc**

**Nicky Ferguson**

**Clax Ltd**

**Contact**

Nicky Ferguson

[sqe@clax.co.uk](mailto:sqe@clax.co.uk)

# Keywords

---

---

distributed learning, e-learning, learning software, VLE, MLE, learning environment, learning tools, software quality, evaluator, adult and community, further education, higher education, JISC, assessment, web services

# Acknowledgements

---

---

We are very grateful to all those who gave up time to help us write this report. Vital to this work were the people we interviewed; JISC programme staff, particularly Richard McKenna who maintained close contact at key times in the evaluation process; and the project staff who responded to our enquiries and made their software, code, and documentation available to us.

# Contents

---

---

<b>1.</b>	<b>Executive summary .....</b>	<b>2</b>
<b>2.</b>	<b>Introduction .....</b>	<b>4</b>
<b>3.</b>	<b>Target audience .....</b>	<b>6</b>
<b>4.</b>	<b>Methodology .....</b>	<b>6</b>
<b>5.</b>	<b>Criteria used in evaluation .....</b>	<b>7</b>
<b>6.</b>	<b>Key points arising from interviews .....</b>	<b>9</b>
<b>7.</b>	<b>Key points arising from testing and checking .....</b>	<b>11</b>
<b>8.</b>	<b>Examples of best practice .....</b>	<b>13</b>
<b>9.</b>	<b>Recommendations .....</b>	<b>14</b>
<b>10.</b>	<b>Individual project evaluation summaries.....</b>	<b>17</b>
	<b>Appendix 1 – Abbreviations and glossary.....</b>	<b>49</b>
	<b>Appendix 2 - The 22 projects and their websites .....</b>	<b>50</b>
	<b>Appendix 3 - Short author biographies .....</b>	<b>51</b>
	<b>Appendix 4 - JISC's Layered Services Framework .....</b>	<b>53</b>

---

---

# Software quality evaluator final report

---

---

Authors Nicky Ferguson, John Harris, Gill Osguthorpe, William Pellett, Suzi Wells

Publication date 6 May 2005

This report is confidential and only intended for the audience defined in section 3.

---

---

## I. Executive summary

---

---

This study, commissioned by the UK Joint Information Systems Committee (JISC), aims to evaluate the software produced by the 22 Distributed e-Learning Tools projects and, where relevant, the effect of the programme's approach and management on the production of the software.

The activities conducted during this study included an online questionnaire, face to face interviews with every project conducted at their lead institution, a selective check of the code created by each project, testing the software products where appropriate, and evaluating the accessibility and usability of the products where appropriate.

A two-page public report and summary of this activity is available at: <http://www.therightplace.net/sqe/>.

### I.1 Findings

#### I.1.1 Code

In terms of general structure and efficiency, the quality of the code we have examined has been high. The main problem areas are in the consistency of coding standards when multiple programmers have contributed, and the general quality of code commenting. These have a significant impact on a third party developer's ability to maintain and extend the code; we consider this a high-risk issue given the open-source nature of the programs. While many projects are excellent in all respects, we feel that some do require minor levels of rework, with a small number requiring significant levels of rework, in order to maximise the programs' chances of success and longevity as active, open-source projects.

#### I.1.2 Software

In terms of general robustness, the overall quality of the software has been high. Many of the issues we have identified have been cosmetic rather than functional.

There have been a number of issues that have made some of the projects difficult to test. Often the installation instructions have lacked sufficient detail. Occasionally it was difficult to tell if the product was functioning as intended because, for example, it was in the process of being moved from one system to another. However, in terms of user instruction, about 50% were good and 25% were excellent.

#### I.1.3 Accessibility

We understand that compliance with accessibility standards was not a key focus for this group of projects and that many have specific target audiences. However, if the projects are to be taken up more widely, there should be consideration of JISC's accessibility requirements. There is a need for academic institutions to ensure that services are accessible to the widest possible number of people, regardless of any special access needs.

#### I.1.4 Usability

Four projects have produced good, usable applications and two more have produced demonstrations that would be very usable if implemented. Of the remaining interfaces tested, four were adequate and six were not adequate. For the remaining six projects, testing was of limited importance or not relevant (although brief usability reports have been produced for two of them).

Poor usability does not necessarily reflect a failure of the project, as many were proof-of-concept. Demonstrating that the application could work and creating a solid code-base were sometimes given higher priority than usability testing and refining the interface.

Most projects had thought about usability, and many of the projects with poor or only adequate interfaces were aware that there were problems to be addressed.

## **1.2 Key recommendations**

The full recommendations from this study appear in section 9. Key recommendations include:

- Mandating all development projects funded by JISC to provide a single point of access to project information.
- Approving the quality-focussed management approach of this programme and clarifying and modifying the reporting document templates supplied so that they can be used in future programmes.
- Providing a short briefing document for senior management on the benefits of open-source (OS) software development at their institutions.
- Investigating the provision of a central (backed-up) version control archive for development projects, using software such as cvs or (its supposed successor) Subversion.
- Encouraging critical peer testing and usability testing throughout software development projects.
- Mandating projects to provide a JISC log-in and well-documented access and installation instructions from an early stage in the project.

## **1.3 Points of interest**

It may be worth making some brief points about three related areas:

### **1.3.1 Development methodologies**

Many, if not all, of the projects are employing rapid application development or agile project management methods. The exact method used seems to have been less important than whether the team are both competent and happy with the approach. Some projects expressed concerns to us about how well the model of work packages and quality planning sits with rapid application development. If the programme manager and the evaluation team are sensitive to the potential problems and aware that aspects of the project's deliverables may change over time, we feel that this should not present a problem. Indeed, many of these projects seemed very successfully to combine rapid development with useful, concise, and accurate reporting.

### **1.3.2 Successful software development**

Although it is difficult to say exactly what makes for a successful project, attributes of the most successful projects we examined included: regular testing; open flow of information within the team and with outsiders, particularly potential users; clear communication; a self-critical eye; quality control; regular management overview and updates; close contacts with other projects and standards bodies; enthusiasm; and, of course, competence and expertise. Undeniably luck comes into it too, but one can insure against bad luck to a certain extent by good planning (see next point).

### **1.3.3 Value of software quality planning and evaluation**

Completing the quality plan early in the process ensured that projects considered software quality and the technologies and standards that they were going to apply from the very start. Many teams mentioned that the quality plan provided the focus that is so important for such short projects. Others found it useful to revisit their quality plans during the project to take stock and, once again, focus. Several teams said they would have planned for software quality without JISC's intervention, but that the quality planning process was easy for them "because we were doing all this stuff anyway". By contrast, a small minority of teams complained that the quality process was burdensome and time-consuming. In our view, the light touch of the quality planning and evaluation required by JISC ensured that all projects had considered the basic issues at an early stage in their development, while not over-burdening them with bureaucracy.

## 2. Introduction

---

---

### 2.1 The study and its background

Twenty-two Distributed e-Learning Tools Projects have been funded as part of JISC's Distributed e-Learning Programme. These projects were due to begin development in September 2004 and complete by March 2005. Each project aims to develop open source software that will be freely available to the academic community. The range of software under development covers accessibility, assessment, collaboration, learning design, mobile, multimedia, person development profiles, and personal learning environments.

The software developed by e-Learning Tools Projects should conform to Open Standards to provide a high level of interoperability. This should allow the software to be easily integrated into existing learning environments. The e-Learning Tools development also conforms with the ELF (e-Learning Framework), a service-oriented factoring of the core services required to support e-Learning applications, portals, and other user agents. The current state of the e-Learning Framework and details of the service definitions can be found at <http://www.cetis.ac.uk:8080/frameworks>.

This evaluation study was commissioned to examine the software outputs from the 22 e-Learning Tools Projects and to make suggestions on management, mechanisms, and evaluation strategies for future programmes. Unusual features within the programme include the short-term funding for these 22 projects, the formal requirements for open source code and compliance to Open Standards, and the emphasis on quality planning, management, and continuous quality control.

### 2.2 The wider background

The HEFCE (Higher Education Funding Council for England) has made IT infrastructure funds available to the JISC to develop technologies to underpin the Funding Council's political and strategic agendas and to work with regional and subject communities to use the technologies to support learning and teaching. The funds will provide a basis for the JISC and the new HE Academy to work together in partnership to achieve shared objectives. This is part of a longer-term plan to implement HEFCE's e-learning strategy.

The aim of developing the Distributed e-Learning technical architecture is to provide better opportunities for learners through the use of better learning tools, easier access to personal learning information such as portfolios, and access to greater quantities of quality assured learning materials.

The aim of defining the Distributed e-Learning architecture for teachers is to provide guidance on how to access, plan, and use e-learning resources within appropriate e-learning systems. This will include guidance on locating and using existing learning resources and advice on how to share teaching materials with others. Difficult issues such as intellectual property rights will be explored and guidance provided. Case studies of good practice in e-learning will be shared, which will include approaches from different subject disciplines.

The Distributed e-Learning Programme will offer benefits to institutions by enabling links between schools, colleges, and universities that can be used to encourage progression into higher education for some institutions. The programme will make available some open source e-learning tools that will complement commercially-provided resources. Provision of tools for personal development planning (PDP) and e-portfolios will also help universities that need to meet the UUK (Universities UK) requirement for provision of PDPs by autumn 2005.

The programme will offer benefits to funders by providing exemplars of good practice in the use of e-learning systems; by exploring how technology can support widening participation and regional partnerships; and by demonstrating how the HE Academy and the JISC can work fruitfully in partnership.

### 2.3 Terminology and definitions in this study

We speak throughout this document of the UK academic community. By this we mean the community who are exposed to JISC-funded projects, services, and resources: Adult and Community Learning (ACL), Further Education (FE), and Higher Education (HE). JISC's remit does not currently cover schools, so when we speak of the UK academic community we exclude schools for the purpose of this report.

Technology and education between them have spawned their fair share of technical terms and abbreviations; we have attempted to mitigate the worst effects of this by ensuring that each abbreviation is expanded at least once in the text and also listed in Appendix I.

## 2.4 Funding: resources and comparisons

It is outside the scope of our work to comment on value for money and we were not given access to all the projects' budgets. However, it is a factor to bear in mind that some projects received several times as much total budget as others. One might expect this to be reflected in the quantity of the deliverables, including documentation. The purpose of this evaluation is to assess how well the projects met their own aspirations and targets, not to conduct a beauty contest between them. For these reasons we have not created league tables or other such devices. There are sections in the appropriate places describing generic conclusions and issues.

## 2.5 Ambitions and achievements – blue sky or down to earth?

There is a wide variety of projects within the 22 evaluated here. Some are attempting to help define as yet inchoate interoperability specifications, some to create elements in a constantly shifting landscape. It is in the nature of experimental development that some will find that they have created something interesting and worthwhile only to find that it has been left in a backwater by other global developments. Other projects have far more tightly defined tasks, such as creating an add-on or an extra piece of functionality to an already well-defined piece of software. Again this makes comparisons difficult and emphasises that the focus should be on how well the project has approached what it is doing and how well it has kept in touch with the quality and the reality of what it is achieving, rather than comparing outcomes at this stage.

## 2.6 Vision and reality in the ELF

A number of projects drew our attention to the gulf between vision and reality in the e-Learning Framework. There is no doubt that it is very useful, probably essential, to have a clear visual representation of the landscape we are seeking to create and encourage projects to operate within. It is both inspirational and informative. Nevertheless, a number of projects said they were disappointed to find out how much of the ELF was still only vision and how few of the bricks existed and could be incorporated in or tested against their applications. It seems that it is too easy to attend a day of PowerPoint presentations and come away with the impression that the diagrams on the screen have a concrete existence. While recognising the critical importance of the vision, it may be worth emphasising that, at this stage, many of its elements are still not realised.

## 2.7 Future success and use

It is too early to say which, if any, of these projects will survive in the medium term and prove useful to the academic or wider community. These things are notoriously difficult to predict and often depend on circumstances beyond all our control. Nevertheless, it is possible to say that well-organised projects issuing excellent documentation, well-commented, maintainable, and extendible code, and powerful, efficient, and usable software will be in a far better position to take advantage of those circumstances.

## 2.8 Attitudes towards the programme and our evaluation

Most of the projects we interviewed welcomed the emphasis of the programme on quality control and regarded the mechanisms such as the quality plan template as useful and lightweight. Similarly they welcomed us, cooperated with us, and showed interest in our work. A small minority of projects, though, seemed to resent both what they regarded as unnecessary bureaucracy and unnecessary evaluation: *"We've got a good track record – why can't JISC let us get on with the real work"*.

## 2.9 Attitudes towards JISC funding

Similarly, some projects welcomed JISC funding, allowing them to improve and extend work that they already wanted to do, while others resented the fact that important parts of their applications had been cut and did not see, for example, why they should use their hardware resources for testing, when these resources were not supplied by JISC. Perhaps this indicates that JISC should make the "partnership" aspect of their funding clearer at the proposal stage. It may be an indication that JISC should be looking to fund fewer projects but to allow real overhead costs.

## 2.10 Limitations of evaluation resources

Our task was to develop a methodology and evaluate 22 very different, innovative, and sometimes technically quite complex projects in a very short space of time. We typically had a little over a day to conduct code checking and software testing and a little over half a day to look at the project's usability. Clearly this is a very short time to

devote to projects some of which have many different parts. So our conclusions are more impressionistic than comprehensive.

## 2.1.1 Limitations of time and schedule

The projects themselves were operating to a tight timescale, with many of them complaining that six months was too short a period for them to accomplish what they desired. Some projects were affected by illness or staff problems, which further restricted them. It may be that a formal time contingency element in each project would help this problem; or it may be that JISC's flexible approach to projects suffering from such problems effectively acts as a contingency without a formal statement.

Clearly, from an evaluation point of view, it is better to conduct a summative evaluation when the projects have finished! That was not possible in this instance, and many of our access and communication problems were doubtless explained by our having to report at precisely the time when the projects themselves were rushing to meet their own final deadlines. For future software evaluations it may be worth considering including both formative and summative elements, e.g. assessing code, usability, accessibility, and documentation at a half or two thirds way stage so that projects can use the feedback and then testing and assessing the finished software a short time after the project has finished.

## 3. Target audience

---

---

We expect the primary audience for this report to be JISC programme and policy staff. Elements of the report will be of interest to the projects concerned and portfolios of project-specific evaluation materials will also be made available to the programme manager.

A short summary of this report aimed at a wider public audience is published at:

<http://www.therightplace.plus.com/sqe/>.

## 4. Methodology

---

---

### 4.1 Pilot interviews

Early interviews were conducted with three projects on a pilot basis to allow us to refine our interview structure and to develop and modify our explicit methodology.

### 4.2 Detailed Evaluation Criteria document

In close consultation with JISC programme staff, we drafted, amended, and agreed this document, which defines the software evaluation criteria to be used to evaluate the 22 DeLeTools projects. This document, which is available on the JISC website, is closely related to the accompanying Evaluation Methods and Tools document mentioned below: the criteria document describes the tasks to be done and what criteria we use to approach them, and the methods document describes how we should do it.

### 4.3 Detailed Evaluation Methods and Tools document

Similarly, we prepared this document, which details the methods used to assess, test, and evaluate the software outputs of the 22 projects in the DeLeTools strand of the JISC e-Learning Programme. Using the Software Quality Evaluation Criteria Document as a high level guide, the Detailed Evaluation Methods and Tools document outlines the specific methods proposed and references the criteria that these methods address. The latter document states:

*Our approach is based not only on the quality standards to which JISC require the 22 projects to adhere and the individual standards which have been identified by the projects themselves, but also the maintainability, extendibility and the general robustness of the 22 projects' code and software. Code review will be a key focus of the evaluation of each project. Other evaluation and testing methods will vary according to the project and its outputs.*

### 4.4 Summary of methods used

We used the following methods:

1. Brief on-line survey/questionnaire (template appears in Appendix).
2. Face to face interview with project staff and follow up by email and phone (full accounts of interviews appear in Appendix).
3. Selective checking of sections of code (per-project summary of code checking results appears below).
4. Lab testing the software on various platforms and in various conditions (per-project summary of test results appears below – full results have been made available to projects and to programme staff).
5. Usability walkthrough and application of heuristics (per-project summary of usability results appears below – full results have been made available to projects and to programme staff).
6. Brief and selective desk audit of project documentation, version tracking, and testing procedures (the results of the audit were fed into the software evaluation and gave the testers insight into which areas of the tool might need particular attention, also alerting the testing team to any issues and bugs prior to testing and informing some project-specific questions asked during the interviews).

## **5. Criteria used in evaluation**

---

---

The criteria used in the evaluation and a brief summary of the measures or "questions asked" appears below.

### **5.1 Match of actual software output to planned output**

#### **5.1.1 Measures**

Are the outputs materially different from those planned? Are they so delayed that the project will not, or is unlikely to, deliver those outputs?

### **5.2 Use of quality plan**

#### **5.2.1 Measures**

Has the quality plan been completed and followed? Has it been used to aid implementing the JISC Software Quality Assurance Policy? Has it been updated? Has it been used in creative and unforeseen ways?

### **5.3 Compliance with the JISC Open Source Policy**

#### **5.3.1 Measures**

The (draft) JISC Open Source Policy (May 2004) itself details the necessary areas of compliance.

### **5.4 Compliance with Open Standards**

#### **5.4.1 Measures**

Have the Open Standards outlined in the quality plan been used? If not, have other Open Standards been used?

### **5.5 Quality control procedures**

#### **5.5.1 Measures**

Are there any documented formal or informal quality control procedures? Is there evidence of them being used? Can we reproduce testing procedures? Are issues and changes being documented?

### **5.6 Project specific documentation**

#### **5.6.1 Measures**

In addition to the project plan and quality plan, we asked each project to give us access to:

- the technical specification of their product
- a summary of their testing procedures
- a copy of their test plan
- reports from version tracking software
- code documentation.

It was anticipated that there would be little or no end-user documentation, and that few projects would have a complete technical specification document.

## 5.7 Quality of software product(s)

### 5.7.1 Measures

**Maintainability and extendibility.** We asked what the software will require to keep it going in the medium term. We recognised that projects will not necessarily want to provide a support service for their software – will other people be able to maintain it? Do they have any plans for a structure to enable this (e.g. deposit at SourceForge)? Similarly, will the code be extendible by others as well as the team?

We followed up the answers to these questions by examining random or selected samples of code to check for commenting and good documentation, which will allow code to be developed, extended, and/or maintained by others in the future.

**Robustness and stress testing.** We tested whether each software product installs reliably and produces repeatable and reliable results when under normal and/or abnormal operating conditions. Different approaches to testing robustness were used as appropriate:

- Compatibility – testing against a range of hardware and software identified to be likely combinations for potential users to deploy. This also covers network compatibility testing, e.g. RM CC3 networks, as well as bandwidth and page weight testing where applicable.
- Standards compliance – see above.
- Test scripts – built on the different potential user inputs the program accepts, scripts are used to discover if all expected outcomes of any action made by the user are correct to the design of the application.
- Destructive testing – a destructive test script is used that covers user input properties, exploring allowable data sets and character types, filling the file system to capacity, forcing media into unexpected states, using invalid file names, corrupting saved files, and injecting faults that might realistically happen in the field using specialist fault-finding software.
- Stress testing – like destructive testing, stress testing applies to applications which have been designed for multiple concurrent users. Stress testing involves multiple users or mimics multiple users accessing the application at the same time.
- Test plan/process review – consult and advise developers on testing procedures, which will suit their needs. Analysis of the quality, appropriateness, and depth of a developer's testing process is the first step in building and releasing robust, dependable, bug-free software.

**Usability.** We attempted to run each software product available to check that it is usable by the target audience. Project staff were asked whether they had any reservations about usability and we investigated any problems of which they were aware. Where software was not available we sometimes followed up answers to these questions by examining documentation to check use requirements were specified and user scenario(s) were created/discussed. Where it was relevant, each project was evaluated by an experienced programmer/usability consultant against a set of general usability heuristics. It was anticipated that the usability of some software would require little or no time to evaluate, since the user interface would be limited or non-existent, and that other projects may require more user interaction and hence more time to evaluate.

**Accessibility.** Accessibility testing largely depended on the type of program, application, or web-based application. In some cases, standard accessibility testing was applied, such as testing against IMS guidelines, WAI and other W3C standards or use of the LIFT tool for example. In some cases screen reader technology compatibility was conducted. In the case of applications with no GUI, accessibility testing was limited to the use of exploratory tools.

## 5.8 Lessons learnt

We talked to each of the project teams about their experience of the QA process, asked for their suggestions, and considered whether compliance with the process had or was likely to have any effect on the quality of deliverables.

We asked each project how it viewed the programme. Answers were fed back to JISC staff on a per-project basis and no evaluation criteria were used on this information.

## 5.9 Future plans

We also asked each project about future plans. Answers were fed back to JISC staff on a per-project basis and no evaluation criteria were used on this information.

# 6. Key points arising from interviews

---

---

## 6.1 Single home page/access point

Some projects did not have a clear access point – the home page field on the JISC DeLeTools page is left blank for several projects and many have not used the CETIS web page. Conversely projects often have several different websites; this can be confusing and lead to a searcher missing crucial information which is only available on one of the many websites. This is not only unhelpful when people like us come to evaluate; more importantly it is unhelpful for dissemination to colleagues or outsiders looking at the project with a view to working together or using the software. Where projects have one clear access point (to which there are consistent pointers from the DeLeTools and the CETIS web pages) then it is much clearer. Ideally this access point should have access to a two-paragraph overview of project, news, developers' blog if appropriate, version news and versions control, latest releases of software, support documentation, bug/issue database, contact information etc.

## 6.2 Programme management

There were many favourable comments about the programme management. "Lightweight", "responsive", "supportive", "we felt we were not on our own" were typical comments.

## 6.3 Inter-project communication

The one commonly expressed reservation was that the programme management had not provided enough opportunities for project staff from different projects to meet. Many would have liked more opportunities to get together, and several projects would have welcomed a more proactive approach to this. For example, rather than relying on social contacts at a programme meeting, they would like programme staff to say "you really should be taking to x, y and z".

## 6.4 Amount of documentation required by JISC

Most respondents welcomed the project quality plan, thinking it was useful and, particularly, that it was useful to review it at a set point in the project. The majority of projects thought that the Quality Plan template, the Project Plan template and the Workpackages template were all a sensible length and that Richard made very reasonable demands. Several projects thought that the process was going to more demanding and bureaucratic than in fact it turned out, and they welcomed this. Some projects that had their first try at filling in the documents returned for more detail said that, although they were irritated at the time, this proved beneficial in the long run. There were a minority of projects which mentioned that they were surprised by the amount of documentation they received and that the amount of documentation required from them in the planning phase came as a shock. Some of these projects had not been in receipt of JISC funding before; perhaps some had not conducted a software development project before. There was some confusion and anomalies caused by misunderstandings about exactly which sections of the templates needed filling in. This could easily be remedied.

## 6.5 JISC Project Management Guidelines

... are long enough that many projects don't read them. In general, the projects appreciated the short documents developed specifically for this programme and did not appreciate, and sometimes ignored, long generic documents which seemed to be designed for longer projects.

## 6.6 Sample documentation

Some projects said they would have liked to see exemplar quality plans. Another had a very inexperienced team and could have used advice about what should go into a specification document. Sample documents for all stages (so long as they aren't seen as more requirements) could help.

## 6.7 Advice on open-source licence

Several projects were confused about the implications of the various licences. They would appreciate guidance from JISC, such as a set of recommended licences plus implications. Some projects said they need to use a less viral restrictive licence than L/GPL (e.g. BSD?).

Several projects mentioned the desirability of having a short briefing paper for institution senior management on open-source software – something to answer the questions: "Why on earth should we get involved in a project where we are giving away all our IP? What's in it for us?"

There was a variety of attitudes from the projects to open-source development, ranging from reluctant acceptance to enthusiastic embracing.

## 6.8 SourceForge

A couple of projects mentioned that they got turned down or ignored by SourceForge at first, or that there was a long delay before SourceForge gave them a space. Others mentioned that there is no guaranteed archiving facility at SourceForge and that some projects have lost code there. It seems that SourceForge works best as a repository for release code but not as a working development tool. This implies that projects must have access to their own version control and archiving tools – and some projects suggested that it would be helpful if JISC were to make a recommendation or, preferably, run a central, archived service, based on Subversion or cvs or similar.

## 6.9 Moodle

The projects we encountered that have worked with Moodle all have very positive things to say about it.

## 6.10 Planning for slack time

Most projects felt that the time allowed for the projects was too short. Especially where projects experienced delays at the start, and also because of Christmas and Easter holidays, they felt that in fact they had more like five months. Several projects expressed the opinion that "The money was fine but we could have done with another three months to spend it in." Others felt that both the timescale and the budget were too tight.

A number of projects have had delays because someone was off ill and there was no slack in the schedule. They felt that project plans should have some slack or contingency time built into them (or JISC should expect a large number of projects to deliver late).

## 6.11 Documentation

When assessing the quality of the documentation, we considered how thoroughly the users of the system and the project requirements were described, as well as the design and system documentation itself. Because of the open-source nature of the projects, we also considered how easily the documentation could be located and used by future developers. With notable exceptions, the documentation was quite widely distributed and it was difficult to determine whether we had located all of the relevant documentation. SourceForge was only used to store documentation by around 10% of projects. Around 25% of projects had very well organised documentation. We found that the user documentation tended to be brief, with only around 15% of projects giving a detailed description of users. Requirements documentation also tended to be rather brief and high-level. In around 25% of cases the requirements were good, with the top 10% giving an excellent level of detail. System and design documentation was the strongest area, and JISC's documentation standard of choice, UML, was used in around 50-60% of cases. Around 75% of projects provided adequate, good, or excellent system documentation.

## 6.12 Suggestions from single projects

### 6.12.1 Microsoft to Open Office/xml conversion

From VLMA: JISC could consider a generic service converting Microsoft Office to Open Office and vice versa and also xml to Open Office and vice versa.

### 6.12.2 System administrators and software testing

From ASAP: "A problem that has been overcome for this project but may arise for others – [our own institution's] systems administrators weren't happy about running software [that our] project had developed. The team had to suggest to the administrators how to test it. The administrators only wanted modules approved by the company that make Blackboard."

### 6.12.3 Marketing the results of the strand of development

From eLaws: "Funding helps, but what would also be useful would be support, marketing, and disseminating the software. Someone could work with HE/FE institutions to show them how they can use the software from this strand of development – cooperative rather than competitive strategies (like Web CT, etc) – there's a gap in the market for this, but only for the moment."

### 6.12.4 Hosting for schools/colleges

From TIP: Strongly suggest that JISC host some services for schools and smaller FE colleges (e.g. TOIA is already being hosted by Glasgow). Schools are keen to use things like LAMS and Bodington but don't have the resources to do it themselves.

## 7. Key points arising from testing and checking

---

---

Testing by EpiCentre has focused on two key activities – the reviewing of code and the testing of functionality. In undertaking this work we have taken into account that these products are not designed for immediate publication as 'finished' products. The projects vary widely in type – some are proofs of concept, some are finished tools, others are discrete bundles of code designed to address a particular piece of functionality. As such, our approach has been more tolerant of errors and imperfections than the regime we would apply to products designed for commercial release. Also, our approach was dependent upon receiving appropriate documentation from the projects, such as installation and user instructions; as a result, where we have identified deficiencies in the quality of a product, it may reflect our lack of a complete understanding because of insufficient or inadequate documentation.

### 7.1 Code reviews

In assessing the quality of the code we have examined, we have addressed the following questions:

- How well is the code structured?
- How efficient is the code?
- How readable is the code?
- How clear are the comments in the code?
- How do the answers to these questions affect the extensibility and maintainability of the code?

In terms of general structure and efficiency, the quality of the code we have examined has been high. The main problem areas are in the consistency of coding standards when multiple programmers have contributed, and the general quality of code commenting. These have a significant impact on a third party developer's ability to maintain and extend the code; we consider this a high-risk issue given the open-source nature of the programs. While many projects are excellent in all respects, we feel that some do require minor levels of rework, with a small number requiring significant levels of rework, in order to maximise the programs' chances of success and longevity as active, open-source projects.

## 7.2 Test reviews

### 7.2.1 Robustness

In terms of general robustness, the overall quality of the software has been high. About half of the projects have followed industry best practice in terms of the use of user evaluations, test plans, and user evaluations. Our experience would suggest that around 75% of the projects are in a pre-gold state, i.e. close to publishable. Many of the issues we have identified have been cosmetic rather than functional.

There have been a number of issues that have made some of the projects difficult to test. Often the installation instructions have lacked sufficient detail (see below for more on installation issues). Occasionally it was difficult to tell if the product was functioning as intended because, for example, it was in the process of being moved from one system to another. However, in terms of user instruction, about 50% were good and 25% were excellent.

### 7.2.2 Accessibility

Compliance with accessibility standards has been one of the main issues we have identified. Across the projects there has been little or no focus on JISC's accessibility requirements. For example, a number of web-based projects could easily have been designed to meet the accessibility requirements more effectively than they do. On occasion this lack of concern for accessibility has been because the system that is being built upon is not, in itself, accessible. There has also been little focus on accessibility in terms of usability and user testing. This is surprising given the clear need for academic institutions to ensure that services are accessible to the widest possible number of people, regardless of any special access needs.

### 7.2.3 Compatibility

A number of the projects are compatible with a wide range of operating systems and browsers. Some of the Java-based applets do not work with, say, MacOS or Linux. This is surprising given the platform-independent nature of Java. Some of the projects have been given explicit agreement by JISC to use languages that may cause issues with non-Windows-based platforms.

### 7.2.4 Installation

In terms of installation, a number of projects have been overly complicated and, as a result, difficult to configure. If third party developers are to be able to pick up code from an open-source repository such as SourceForge, then these projects would benefit from clear installation instructions and a simpler installation process. A poor installation process can add a significant time overhead and will deter developers from making use of code that may in fact be very useful to them.

## 7.3 Usability testing

With reference to usability, four projects have produced good, usable applications and two more have produced demonstrations that would be very usable if implemented. Of the remaining interfaces tested, four were adequate, and six were not adequate. For the remaining six projects, testing was of limited importance or not relevant (although brief usability reports had been produced for two of them).

Poor usability does not necessarily reflect a failure of the project, as many were proof-of-concept. Demonstrating that the application could work and creating a solid code-base were sometimes given higher priority than usability testing and refining the interface.

We were not in a position to evaluate the pedagogical value or the uniqueness of the features offered by the applications. Either of these factors might be considered more important than poor usability, although usability should be considered by all application developers where the application has human end users.

Most projects had thought about usability and many of the projects with poor or only adequate interfaces were aware that there were problems to be addressed.

Projects undertaking their own usability testing did not always produce the most usable interfaces. This may be because projects did not have enough time to implement the changes needed, or because the testing was not sufficiently well-designed, critical, or independent.

## 8. Examples of best practice

---

---

**Interactive Logbook** have a good system for logging actions and decisions, as well as a complete QA system for projects that they are hoping to market. <http://portal.cetadl.bham.ac.uk/ilogbook/>

**Serving Maths** seem to have developed a good open-source community that will maintain the project in the future. This may be down to the enthusiasm and organisation of the project lead, and possibly to the attitude of maths departments. The project's active use of bug tracking software and informative project website are also exemplary. <http://mantis.york.ac.uk/moodle/course/view.php?id=2> . [http://mantis.york.ac.uk/serving\\_maths/](http://mantis.york.ac.uk/serving_maths/)

**WCKER** have a good open-source wiki integrating version tracking, which they use for their project website. Other projects manage to do a similar thing with a hand-built web page, but there are definite advantages in terms of ease of use and multiple authoring. The WCKER site is at: <http://waterbuck.conted.ox.ac.uk/cgi-bin/trac.cgi> and the home page for trac, the open source tool they use, is at: <http://projects.edgewall.com/trac/>.

The **Open Mentor** team undertook, documented, and made available regular usability testing with real users. Doubtless as a result of this, their application scored very highly for usability. We recognise that not all projects were in a position to conduct usability testing, but this confirms that for any application which is aimed at a variety of end users, usability testing is an essential component. [J.L.Rae@open.ac.uk](mailto:J.L.Rae@open.ac.uk)

The **RAMBLE** website is simple, well-written, and clear. It is well-designed without being over-designed. <http://ramble.oucs.ox.ac.uk/>

**Manchester PLE/VLE Framework** has exceptional commenting, formatting and naming conventions, and the code was organised logically with excellent error handling. Clearly a lot of time and care has gone into writing this code and the result is a professionally developed, highly extensible, maintainable program.

**Assis** have presented project documentation in a number of clear, well written and concise work packages. The documents provide detailed information on the design of the project's software and are available in PDF format. [http://www.hull.ac.uk/esig/assis\\_deliverables.html](http://www.hull.ac.uk/esig/assis_deliverables.html)

## 9. Recommendations

---

---

### 9.1 Future programmes

#### 9.1.1 Mandate single access point, recommend wiki or similar tool

JISC to mandate that projects should have a single access point for information (naturally, some of this information may be duplicated in other places, such as CETIS and DeLeTools pages, but there should be consistent pointers from there to the single authoritative source). At the minimum, this access point should have pointers to a two-paragraph summary of the project, regularly updated brief news pages, documentation, version tracking, and bug and issue reports. JISC to recommend an open-source tool which achieves this, while allowing any solution (including hand-made web pages) which achieves the functionality outlined above, and to provide a template (or web page) containing indicative content, in line with the Quality Plan and other documentation that JISC issues to projects.

#### 9.1.2 Project communication

Ensure inter-project communication takes place in spite of pressure of individual project deadlines. Organise regular programme meetings. Take proactive approach in mandating certain projects to contact each other, in particular at the beginning of the project life cycle.

#### 9.1.3 Collaboration

Related to the previous point, some projects were working on such closely related or overlapping tasks and projects that they would have benefited from direct collaboration during the development process. More collaboration may have led to new results or synergistic advantages or just to avoiding duplication of effort. Perhaps, rather than "forced marriages" projects could be encouraged to work together through practical work sessions and brains trusts (if this suggestion is examined further, it is important that these sessions are not just processions of PowerPoint presentations but practical hands-on sessions involving developers and the project teams).

#### 9.1.4 Project documentation

Minor edits should be made to current Quality Plan, Project Plan, and Workpackage templates to ensure clarity and integral instructions on which sections to fill in and which not. Maintain short documents and quick responses where possible. Look at other documents which are dense and/or long to see if summaries can be made. Provide exemplars where possible and desirable.

The Quality Plan has been a very useful innovation. It should be used as a review tool with projects and JISC programme staff at least once during the life of the project.

#### 9.1.5 Open-source guidance

##### 9.1.5.1 Senior management briefing

Provide short briefing document for senior management on the benefits of open-source software development at their institutions.

##### 9.1.5.2 Guidance on licences

JISC to recommend one or a selection of open-source licences. If this is considered not possible or desirable, then to produce a short summary of different licences and their features and applicability.

##### 9.1.5.3 Creative Commons

In the context of examining licensing, JISC should also look at the relevance, for projects producing content, public reports, and documentation, of the Creative Commons UK work and the licences which they are producing (still in draft).

<http://creativecommons.org/worldwide/uk/>

### 9.1.6 SourceForge and version control

Clarify that JISC's mandate to projects is to deposit release version of software at SourceForge, not necessarily to use it as a version control archive.

JISC to investigate providing a central (backed-up) version control archive for development projects using software such as cvs or (its supposed successor) Subversion. If not feasible, then JISC to issue recommendations to projects about local development version control in addition to the SourceForge release mandate mentioned above.

Wherever projects deposit their release versions, it is essential that they are accompanied by adequate technical documentation, including installation and usage instructions.

### 9.1.7 Templates for projects

Ensure the Quality Plan, Project Plan, and Workpackage templates are short, clear, and unambiguous, so that they are clear and relatively easy to fill in and keep up to date. Of course, the projects will not find them easy to fill in if thought has not been given to the appropriate issues – but then that is the point. JISC should consider providing brief filled examples of good practice:

- Quality Plan
- Project Plan
- Workpackage summary
- Test Plan
- Functional Specification
- Installation and Usage Instructions.

Newcomers should be allowed to ask questions such as "What is a test plan?" in a non-judgemental environment.

The Workpackage template needs amending and JISC might also consider issuing two versions of the Project Plan: a reduced version for smaller projects and a complete version for larger projects; this would encourage smaller projects to complete the document and, potentially, apply better project management techniques.

### 9.1.8 Usability testing

Usability testing is often considered expensive and is frequently cut from development budgets. However, as others have pointed out before us, it is more expensive to spend your development budget building an application which no-one will ever use. Where an application is to be used by end users and not just machines, then high-quality, critical usability testing should be considered an essential part of a project's budget and not a luxury.

### 9.1.9 Clarification on mandating open source

Some projects don't seem to believe in, or perhaps completely understand, open source, saying that they don't mind making their software available to academic institutions but don't want commercial companies using it to make a profit. There is a need for JISC to clarify the rationale for OS (perhaps point projects to the OS-watch pages?) and also to make each project's commitment to OS crystal clear at the beginning of the projects. We got the impression that one or two projects signed up to it because it was mandatory but did not prioritise the OS side of their projects.

We applaud JISC for taking the step to mandate open-source development as part of this programme and we encourage JISC to extend this to all such software development that it funds. JISC should explain clearly to projects at proposal stage the advantages and rationale for this commitment. An added advantage of JISC clearly mandating this for future projects is that it allows projects to put pressure on other developers to make their tools available under open-source licences: "If you want your software used in JISC-funded research, you'll have to make an open-source version available."

### 9.1.10 Encourage regular critical testing

Projects should be encouraged to do regular testing and to use their own code and software in practical applications wherever possible. Regular, self-critical testing and use can make up for many process deficiencies. Any testing is better than no testing - give it to colleagues, peers, other projects, and project partners to install and test, and ask them to keep honest, impolite diaries of their experience.

### **9.1.11 Commenting and code structure**

JISC should specify that good code structure and commenting are an essential part of maintainable and extendible code, and explain clearly at the beginning of projects that it is mandatory and why it is so important for open-source software. Whether projects are tightly defined and practically focussed or more blue sky research projects should not affect their commitment to maintainable and extendible code.

### **9.1.12 Installation and usage instructions**

JISC should specify that installation and usage instructions are an essential part of maintainable and extendible code and explain clearly at the beginning of projects that it is mandatory and why it is so important for open-source software. As evaluators, our biggest problem has been getting certain software working and then working out how to use it with inadequate, poorly prepared or unavailable installation and operation instructions.

### **9.1.13 UML and mandating technology**

Projects were mandated to use UML and a workshop was provided. Many projects did not understand why this was the case and did not use UML or abandoned it, one project describing the UML workshop as "dreadful and poorly taught". From an evaluator's standpoint, if all the projects had used UML it would have made our task easier. However, if projects are to be mandated to use a particular technology, then the rationale needs to be clearly explained and the workshops need to be professionally run to engender enthusiasm amongst their audience.

Some projects might have found that UML activity diagrams (with swim lanes) would help them (and us) to understand the system in context, while UML use-case diagrams might be of more use to people who are producing object-oriented code.

## **9.2 Future evaluations**

Of the recommendations above, several, if implemented, would make a future evaluation much easier to conduct:

- A mandatory single access point to project information (9.1.1)
- Revised Quality Plan, Project Plan, and Workpackage templates, used as a review tool at least once during the life of the project (9.1.4 and 9.1.7)
- Easy access to a central version control archive for development projects, accompanied by adequate technical documentation, including installation and usage instructions (9.1.6)
- Easy access for programme staff and evaluators to the software itself (9.1.11).

### **9.2.1 JISC log-in**

We further recommend that each project should be asked to set up a JISC log-in (username and password) at an early stage in the project, so that when this is required by programme staff or evaluators, it is already in place and does not require negotiation and delays.

### **9.2.2 Monitoring and review**

With a generic methodology and the above recommendations in place, much of the monitoring and interviewing work could be done as part of normal process, by programme staff or freelancers with a restricted brief; JISC should look at whether peer review from within the programme might play a part in this.

### **9.2.3 Testing**

With the above issues dealt with and documented, the code, software, accessibility, and usability testing could all be tightly defined and done by externals on the basis of x days per project, (exact number of days to be defined).

### **9.2.4 Using this system more widely**

With such a framework in place, comparable evaluation could happen across all JISC development programmes producing software and/or services. Interviews, testing, code review etc. could be put out to tender with a definite expectation of the time allowed for each task; thus a small percentage of each programme could be allocated to evaluation with fixed and comparable results.

## 9.2.5 Scheduling evaluation

For future software evaluations it may be worth considering including both formative and summative elements, e.g. assessing code, usability, accessibility, and documentation at a half or two thirds way stage so that projects can use the feedback and then testing and assessing the finished software a short time after the project has finished.

# 10. Individual project evaluation summaries

---

---

## 10.1 AcademicTalk

### 10.1.1 Aims and objectives

- To develop a technically robust and quality interoperable e-learning application - AcademicTalk\_2
- To integrate additional interface features that improve the usability of the existing tool
- To develop a web-based environment supporting, managing, and pedagogically adapting the use of AcademicTalk\_2, making it suitable for different learning contexts.

This project is developing an existing innovation and tool (AcademicTalk) that successfully supports synchronous online educational argumentation for open and distance learners. This current application is being developed into a transportable, re-usable, and adaptable tool that can be used in a range of educational contexts to realise structured, and yet tailored and flexible, pedagogical approaches to dialogical learning. This will lead to benefits for a wide range of learners.

### 10.1.2 Brief comments on software and code delivered

The team has developed a desktop application allowing structured online chat / discussion. A configured Jabber server is used to provide the chat service.

Features of the tool include:

- using sequences of sentence openers (phrases the learner must use, designed to guide thinking along certain lines) to push dialogue in a certain direction (e.g. proposition, expansion, challenge...)
- a chat space for the social aspect of dialogue in addition to the game space for the structured dialogue
- learners must reply to a particular message
- the "local window" shows the sequence of the discussion, encouraging a response to the whole argument
- the ability to generate a summary of the dialogue, which might for example be used as an essay plan
- the ability to structure activities, e.g. a page of readings, followed by the learner raising comments from different texts, then discussion of wider issues, then debate, then summary and reflection.

The project team provided all necessary code and the client side software, which can be considered to have met many of its objectives. We were not able to assess the server side software.

The code is considered maintainable and extensible; however, there were some inconsistencies.

The software does have accessibility issues which will limit and may alienate some potential users.

### 10.1.3 Summary of usability testing

The application is generally engaging and usable.

The discussion feature has some non-intuitive aspects and would benefit from additional usability work.

### 10.1.4 Summary of code checking

Commenting was not entirely consistent, with creation date and history often omitted. In general, variable, class, and function names were descriptive, with class names adopting the standard nomenclature. However, naming conventions were not consistent across classes, with different authors applying different conventions. It is good practice if all programmers adhere to the same set of standards. The code has been structured in a logical manner. Good use has been made of the standard features of Object Oriented Design methodologies. Nothing was found to

suggest inefficiencies in the code, and the degree to which the code has been commented and the readability are both of a sufficient standard to aid the maintenance of the code.

## **10.1.5 Summary of software testing**

### **10.1.5.1 Robustness and compatibility**

The AcademicTalk program appeared to work as intended. However, there were issues with compatibility, as it requires Java JRE 1.4.02.02 and will only run on Windows 2000 SP4. This prevents it from being backwards compatible or open to users with different operating systems.

However, testing in two locations with multiple users showed AcademicTalk to be a robust program.

### **10.1.5.2 Test documentation**

No test documentation was provided; however, we were made aware that AcademicTalk had been tested extensively.

Installation and user instructions were also not provided at first. During the process of testing, a newer version of the program was made available to us, which was easier to install and run, as well as basic user documentation, which helped us to understand how to use the application.

### **10.1.5.3 Comments**

Overall the AcademicTalk software was successful and fit for purpose. However, we would recommend that changes be made to make it compatible to other operating systems.

### **10.1.5.4 Accessibility**

The user tried to use both Supernova and JAWS screen reader tools to view the AcademicTalk program. A screen reader cannot read the content contained under the page title, a serious drawback. The interface made use of tab keys and the cursor but did not use keyboard shortcuts. There are no issues with magnification tools.

## **10.2 ASAP**

### **10.2.1 Aims and objectives**

The aim of the project is to develop, integrate, and evaluate e-learning tools to support teaching, learning, and assessment of computer programming languages. The specific objectives are:

- the publication of appropriate standards for the incorporation of unit tests in the educational sector
- the publication of tools for authoring and using unit tests, which can be employed in conjunction with a variety of VLEs (virtual learning environments)
- interfaces for relevant third-party software, including the Roboprof and JPlag initiatives.

The principal outcome of the project is a set of application services, made available to the UK higher education community, to automate key components of assessment in the teaching of computer programming languages. These include:

- automatic unit-testing of student code, with associated feedback
- automated generation of objective questions to test learning outcomes
- plagiarism detection and reporting software.

The software is split into application services and user agents, to allow interoperable deployment in learning management systems.

### **10.2.2 Brief comments on software and code delivered**

The outcomes of the project include:

- Blackboard system - automated Java marking system using Blackboard
- uPortal system - automated Java marking system using uPortal
- JAJM - Java Automated Java Marker, a common server for undertaking the actual marking.

The ASAP project provided all the required code and the client side software.

The ASAP project has met its objectives although the quality of the code varies between the three programs.

### **10.2.3 Summary of usability testing**

We tested the interface to JAJM provided through Blackboard. The interface is fairly simple, so only a brief test was required. As many aspects of the interface are provided by Blackboard, there are limits to what can be tested. Design and navigation are therefore not considered.

Usability testing is of limited relevance to this project. The interface is clear and usable, intuitive to use, and provides suitable feedback to users.

Error messages are clear and propose a solution to the problem. When students are submitting assignments, it might be useful to let them know what will happen next. There seem to be some inconsistencies as to what counts as an error, for example whether you are allowed to omit a file.

### **10.2.4 Summary of code checking**

Three modules were reviewed: TAPAS, AJMPortlet, and AJMPortletWS. Commenting is not consistent, varying widely across the three modules. In some cases it is of a good standard; in particular, the classes located in the AJMPortletWS. In other cases commenting is sparse or non-existent.

In general, the classes were found to be concise and on the whole, Class and Method names were descriptive. However, there is poor consistency in naming conventions for variables and some remaining class names. Some files were not well formatted and ten files were found to contain “\todo” comments. There were a number of files that contained hard coded strings - it is good practice to pull these strings out to a global configurable location. Some of the classes output HTML directly - it is not clear without a more detailed analysis of the architecture of the system as to whether or not this would have an impact on maintenance or scalability, but in general it is good practice to make a distinction between programming logic and presentation. With good documentation, detailing the architecture and design of the modules and if the points raised above are addressed, it should be possible for a programmer to maintain the system.

### **10.2.5 Summary of software testing**

#### ***10.2.5.1 Robustness and compatibility***

The user was able to view the web pages on all browsers and operating systems tried. The user was able to alter the assignment that is played to the students. The user was also able to attempt to upload files to the system; however, as the user did not have access to any actual \*.java files the system was unable to compile them. Nevertheless, the system produced the correct error responses to these files.

Both browsers tried on MAC OS x3.5 refused to allow the upload of the user's dummy files. However, the user was not able to try the upload with a genuine \*.java file.

#### ***10.2.5.2 Test documentation***

ASAP provided some testing documentation, although this was not a specific test plan of test scripts.

There were also testing reports submitted.

#### ***10.2.5.3 Comments***

The ASAP project software worked without problems; however, there was minimal supporting documentation on testing.

#### ***10.2.5.4 Accessibility***

The user ran through a typical sequence, making an assessment available as an instructor, viewing the grades, and attempting to submit exercises as a student. We tested for general keyboard navigation accessibility and JAWS screen reader compatibility.

All worked as expected – the user was able to make assessments available as an instructor and to (attempt to) submit exercises as a student (the user couldn't actually submit an exercise, as they must have a valid Java file to submit).

## **10.3 Assis**

### **10.3.1 Aims and objectives**

Assis will provide tools for a teacher to browse, search, preview, and select assessment objects from question banks; these can then be incorporated into packages of content which may also include sequencing instructions. The assessment objects in the content package will contain pointers to an external rendering service, to be developed by Assis, removing the requirement for teachers to have any knowledge of the technical details of how the object is presented to the learner or how the learner's responses are captured. For learners, a player will be produced that will present material from content packages and integrate with the Assis external services for presenting assessment objects and the processing of sequencing rules. Assis aims to provide teachers with the opportunity to design innovative learning activities and sequences of activities incorporating formative assessment, and for learners to access these materials in the manner that suits them best. The suite of open source services and tools that enable these activities will be standards and specifications conformant. The project also aims to provide discrete tools and services that need not be used exclusively by the outcomes of Assis.

### **10.3.2 Brief comments on software and code delivered**

The software delivered by Assis met their aims and objectives as best as we could tell. However, no access was given to the package building area of Assis, which limited our assessment.

Despite having Flash-based content, the Assis software is also considered accessible.

The code delivered was good; however, there were some inconsistencies within it, specifically in regards to the commenting conventions, which varied between multiple authors.

### **10.3.3 Summary of usability testing**

Usability testing is of limited relevance to this project; we tested a Flash demonstrator (not a working piece of software). The demonstrator has a clear, well-designed, and reasonably usable interface, although there are issues that should be addressed. More help and instruction should be provided to the user.

### **10.3.4 Summary of code checking**

There were varying commenting styles and coders did not follow the Javadoc format consistently. Variable, method, and class names were excellent overall. Code written by different people was very inconsistent, for example some coders used C style of braces and some used Java style. Code was well structured into logical folders, with WSDL-generated files kept separate. Good object oriented design and evidence of UML were present.

Some classes were not well commented, appeared unfinished, and used //TODO comments. Some error handling appeared just to send text strings to the command line/application; this doesn't help in a real time application, as exceptions are not adequately dealt with.

### **10.3.5 Summary of software testing**

#### ***10.3.5.1 Robustness and compatibility***

The Assis project can be considered both robust and compatible, working well on a majority of common operating systems, including Windows XP and 2000, Mac OS X.1 onwards, and Linux (Redhat Fedora 3). It also worked without issue in the browsers it was tested within.

Bugs are considered minor and are mainly cosmetic or to do with easily fixed functionality.

#### ***10.3.5.2 Test documentation***

There was no evidence of specific test documentation despite the project as a whole being exceptionally well ordered and presented in work packages on their website.

#### ***10.3.5.3 Comments***

The Assis project was very well presented, with robust software and detailed, high-quality, technical information on their website, although specific testing and user documentation was not present.

#### **10.3.5.4 Accessibility**

The Assis project is a web-based tool, so it is accessible to screen readers such as JAWS. The pages read in a logical order and the tabbing works correctly. In the example tested, the multiple choice answers that were presented in a form and the field for each possible answer were well described, so that the user was able to make the required choice when the fields were displayed in the JAWS Forms Field window.

The [Previous] and [Continue] buttons were presented as part of the same form as the multiple choice answers. The user would have expected the navigation buttons to have been page links rather than form buttons and initially looked for them in the JAWS Links List window. A visually impaired user may have a problem with navigation due to this issue. As only one content package was available for testing, the user does not know if this is a generic issue with Assis or if it is related to the construction of this package alone.

Flash content was used within the package and this was navigable using JAWS without problems. However, it should be noted that other screen readers such as Supernova are not able to read Flash elements and, as such, areas of the package may be completely missed out by visually impaired users using such tools.

No access was given to the package building area of Assis, so the user was unable to test it for accessibility.

## **10.4 Manchester PLE/VLE Framework**

### **10.4.1 Aims and objectives**

The Manchester PLE/VLE Framework is a development that is in part derived from the Bodington VLE development path, and that is under consideration as the basis for future releases of the Bodington e-Learning System. The aim of the project is to provide a modern, well-engineered, and extensible version of the Bodington II system, amenable to use as a personal learning environment (PLE), utilising open source components and open standards wherever possible.

### **10.4.2 Brief comments on software and code delivered**

The Manchester PLE/VLE Framework project provided us with all information requested.

The code is of exceptional quality, with great attention to detail, and we consider that the project is very likely to have met its objectives with high quality deliverables.

We also consider the code to be extremely maintainable and extensible.

However, due to the size of the Manchester PLE/VLE Framework, by no means every feature has been tested.

### **10.4.3 Summary of usability testing**

The end users are developers using the APIs, so a usability walkthrough was not appropriate.

### **10.4.4 Summary of code checking**

Excellent commenting, a huge amount of care and attention has been put into documenting all classes, methods, and variables. All files reviewed were indented to an excellent standard and highly readable. Good naming conventions have been applied to class, method, and variable names, and all classes reviewed were formatted to an exceptional standard. The application appeared to be broken down into logical components and had excellent error handling.

### **10.4.5 Summary of software testing**

The Manchester PLE/VLE Framework project documentation contained previous tests of their back-end system. This evaluation is of their front-end system.

#### **10.4.5.1 Robustness and compatibility**

The pages are laid out well and the user was able to answer each question and add explanatory comments in the edit field. The user was able to save the answers and view/edit them as required.

However, if the user tries to format a reply using the [Enter] key to create a new line/paragraph the [Enter] keypress is ignored, thus causing the layout of the answer to be saved incorrectly.

Also if the user selects a heading such as 'Information Technology' they are taken to a page that simply informs them that there are no questions available: 'There were no QTI questions in the input.' However, there is still a 'Save results' button on this page and if the user clicks on it an 'HTTP 500' error page is returned.

If the user clicks on the 'Save results' button without selecting any of the radio buttons first, an HTTP 500 error is also generated. If the user enters text into the comments box without selecting a radio button and clicks 'Save results' the error page is still generated.

We assume that, apart from the HTTP 500 errors, all is working correctly and that missing content (e.g. that generating QTI errors) is separate from the VLE itself.

The separate code review shows the strength of the Manchester PLE/VLE Framework project.

Overall, the Manchester PLE/VLE Framework is robust and compatible, although there were a number of HTTP 500 errors.

#### **10.4.5.2 Test documentation**

The Manchester PLE/VLE Framework project has produced a large amount of information, which is very easily available from the project wiki.

Although no specific testing plan is available, there are user cases and test results available. This shows clear signs of a QA process tackling different parts of the Manchester PLE/VLE Framework.

#### **10.4.5.3 Comments**

The size and complexity of the Manchester PLE/VLE Framework project meant that the evaluators were not able to test a great deal of the project. However, the project team have clearly put in extensive development work and they have provided clear and concise information regarding the project.

#### **10.4.5.4 Accessibility**

The Manchester PLE/VLE Framework is a web-based tool which uses standard HTML coding that is suitable for accessing using a screen reader. There are no Flash or other elements within it that could potentially cause accessibility problems.

The links are well labelled so that they can be understood when taken out of context of the web page and displayed in the JAWS Links List window.

There is a form on each page for the user to give feedback. However, the radio button fields repeat the question in each description, which means that when the user displays the form fields in the JAWS Form Field window the user has to listen to JAWS reading out the question again before it reads out which response they are currently selecting. It would be quicker for a JAWS user to just have the responses read out.

## **10.5 DELTA**

### **10.5.1 Aims and objectives**

The fundamental aim is to provide an architecture to allow teacher- and learner-controlled materials to be shared, and to "grow context" with sharing. In the case of learners, the idea is to allow work that has been commented on, or even assessed, to be indexed as "tertiary courseware", and shared with other learners. In the case of teachers, the vicarious materials can be case studies of teaching approaches. Both raise the need for a managed system, which allows distributed resources to be submitted, searched, and retrieved, based on standardised meta-data. Extensibility is a key issue, in that it should be possible for new resource types to be added and shared within the learning community. This will be realised by providing a managed meta-data server; this will contain the schemas for the sharable resources, which are then distributed amongst the community. The project aims to deliver a range of server side components based on the following open source tools and standards: W3C OWL, W3C RDF, Protégé, RACER, HP JENA, the W3C RDQL query language, and the W3C SOAP protocol. All code will be developed in Java and open source tools (e.g. JUNIT, Tomcat, etc.) and all of the systems design will be based on the UML specification.

The project is about "developing a generic resource sharing architecture" and encouraging the development of "learner controlled vicarious resources". The resources in question could be any number of things, e.g. profiles, skill sets, question banks, learning objects, or journal articles. The original proposal mentioned using GRID technology, but they have used a Web Services approach instead. *"This ontology-based stuff more naturally uses semantic web technology, not grid technology"*.

## **10.5.2 Brief comments on software and code delivered**

The DELTA project's software has met its objectives and has few errors. Its particular strength lay in its ability to be used across the most common operating systems without issue.

The software had accessibility problems which may limit its potential users.

The code contained little commenting, which is a severe hindrance to any potential developer looking to maintain or build upon the DELTA software.

## **10.5.3 Summary of usability testing**

We tested from the perspective of a user searching for some learning materials, using them, and making a comment about them.

The programme was very slow to use; the project team warned us of this fact. This was at least in part due to the fact that we had loaded the software from the development server. However, it seems that the application is very resource intensive and that this would always have some effect on the usability.

The interface is mostly well presented but it is unnecessarily complicated and slow to use. Rules for completing the form fields need to be indicated and the operation of the form objects needs to be made more consistent with HTML forms. The purpose of the Graphic display (the Java applet) and its relationship with the rest of the application needs explanation.

## **10.5.4 Summary of code checking**

Classes were small and concise. Variable, class, and method names were all descriptive and naming conventions were good. There were no problems with formatting and consistency. Overall the quality of the code was good, but was seriously let down with little or no commenting. All files should start with a comment block, which states the class name, author, creation date, version information, and preferably a history of changes. There were some minor issues, including hard-coded database connection parameters (requiring a recompilation each time the details change, so it is much better to read these in from a separate config file and a number of functions that returned Boolean values when in each case the return value was fixed to return true (not only does this make the return value redundant but more importantly the end user of the class might waste time writing conditional code to check the return value). The lack of comments would hinder maintenance, although the code is sufficiently small to ease the process.

## **10.5.5 Summary of software testing**

### ***10.5.5.1 Robustness and compatibility***

The DELTA program is a web-based program used for applying and sharing documents, links, and other files. There are no problems in operating any commands or using the product. The user noticed that certain searches were taking a long time but produced a correct result.

The cross-compatibility side of the testing was the same as above, but the user had a problem with a Mac running OSX 3.5, where the search and layout function of the product was displaying incorrectly. Other than this, the program worked on all kits tested on and the functions stayed the same and operated correctly.

The user was able to create a user name and password without incident.

All of the resources were accessible, including the wma file that played perfectly through mplayer.

All of the Flash content appeared and functioned correctly.

The user was able to create, view, and delete resources as required.

To summarise, the DELTA project is considered robust and compatible, with only minor issues.

### ***10.5.5.2 Test documentation***

DELTA provided a number of technical documents on their own project website. These included detailed plans and evaluations of the software and a very good test script which, while not a test plan, would catch most functional errors.

### ***10.5.5.3 Comments***

The DELTA project's strengths lay in both the software and the project website, which was easy to use and find information in. The software was robust and compatible on all Windows, Mac, and Linux operating systems.

#### **10.5.5.4 Accessibility**

The DELTA project is Flash-based and therefore has many Flash-related accessibility issues. In this project the only accessible options were three text boxes. Also, the tabbing order is at fault in this program. This means that the program cannot be read using a screen reader and does not meet WAI priority guidelines.

## **10.6 eLaws**

### **10.6.1 Aims and objectives**

The aim of the project is to provide a Web Service interface to the existing server, which allows learners to make and share annotations on existing web pages via an entirely web-based user interface.

The objectives are as follows:

- produce a usable annotation Web Service
- produce an example consumer for the Web Service to provide a client user interface
- produce an open system that can be used by third party developers to provide annotation functionality to their own web-based software.

The software is intended to extend an existing piece of software developed at the University of Huddersfield by providing a Web Service interface to the existing functionality.

### **10.6.2 Brief comments on software and code delivered**

The team have developed an application allowing people to annotate websites. The annotations can be categorised and reports produced.

Outputs include:

- client (service consumer)
- Web service.

The eLaws project provided all their code and client side software.

We consider that the project's objectives have been met.

The code was of varying quality and the software had a number of particular issues.

The software is very accessible.

### **10.6.3 Summary of usability testing**

We tested from the point of view of a user annotating a web page, with the intention that an editor would incorporate some of the notes into the main text at a later date. This involved both annotating and viewing a report of all annotations.

The e-Learning Web Annotation Service is reasonably usable. With a few alterations it would be suitable for use by a wide range of people without any support.

### **10.6.4 Summary of code checking**

Overall, the eLaws Service Source code was well structured. The code looks easily maintainable as it is, but we would suggest providing more documentation and introducing some constants; this would remove some responsibility from the programmer and place it on the compiler, where it belongs.

The annotyClient code had fairly poor file headers and commenting generally, although all files did have author and date created. Formatting was consistent and indenting was good, and the code had a very logical structure and was easy to follow. Removing the hard-coded HTML files could help with code maintenance.

### **10.6.5 Summary of software testing**

#### **10.6.5.1 Robustness and compatibility**

While testing functionality and cross-compatibility across the required Windows and Mac platforms, the user encountered three main recurring bugs.

When attempting to use the software in Linux, the user was unable to log in to the program at all. Clicking on the log-in button would simply reload the page.

#### **10.6.5.2 Test documentation**

The eLaws project provided all the necessary documentation required to install, run, and use their software. From a maintainable and extensible point of view this is very good.

However, there was no specific documentation available on the project team's own testing.

#### **10.6.5.3 Comments**

eLaws could be considered maintainable and extensible, as the project team provided information on how to install and run their program as well as information to potential users on the program's operation.

However, there was no evidence on the project website of work packages or clear use of version control.

#### **10.6.5.4 Accessibility**

The eLaws project is web-based, which makes it accessible with screen readers such as JAWS.

The page reads in a logical order and the tabbing works correctly. The data entry sections make it easy to enter text into fields; these are generally well labelled, so that the JAWS user is able to understand the meaning of the fields when they are listed in the JAWS Form Field List window. The only occurrence of a bug was when the user was trying to log in to the tool using a screen reader.

## **10.7 ePET**

### **10.7.1 Aims and objectives**

This project aims to consolidate and fully document the generic e-portfolio tool and to offer it to the JISC community as a freely available, standards-compliant, open source e-portfolio application.

To enable this, a REpresentational State Transfer (REST) based interface will be developed around the data structures of the e-portfolio to provide Web Service fronted access to data and functionality and all tools comprising the generic e-portfolio will be redeveloped to use this interface.

### **10.7.2 Brief comments on software and code delivered**

Comments on ePET have not been completed due to late submission of code and software.

### **10.7.3 Summary of usability testing**

We tested the profile management and the profile viewing and commenting interfaces available through the demonstration site. We looked at the system from the point of view of a student maintaining their portfolio and of a tutor viewing a student's CV and making comments.

It should be noted that the ePET application was the last to be tested (by more than a week) so they may have had more time to bring finesse to their interface than other projects.

The application is well designed and very usable. However, improvements should be made to the error messages and to the navigation, perhaps by providing breadcrumb navigation.

### **10.7.4 Summary of code checking**

The code sample consisted of a single file sent to us by the developers. They did supply full source code but as the program is only extensible and maintainable within the context of the Zope development environment, the program would be extended through Zope's user interface with only minor tweaks at code level. A review of extensibility and maintainability (the crucial factors we are looking at) should therefore review the technical architecture, design and documentation, a full source code review is inappropriate in this instance. The single file reviewed was well commented and the code was clear and of a good standard.

### **10.7.5 Summary of software testing**

#### **10.7.5.1 Robustness and compatibility**

This is a web-based tool that has three user levels: student, teacher, and admin.

When logging in as a student, the user has the widest range of options to add, edit, and delete entries as required. In general this worked well; however, certain functions caused server error pages to be generated and 'page not found' errors. Also the user attempted to add a user to the system who could view their work, but the new user did not receive a confirmation email containing their user name and password.

The user found that the navigation of the site could be confusing at times, as the tabs along the top of the screen did not retain their highlighted state for the section that the user was in. Often the tabs would revert to having the home tab highlighted or none highlighted at all, regardless of the section the user was currently in.

When logging in as either staff or admin the user still had the same tabs available at the top of the screen. However, only the home tab was active: the others just appeared to refresh the current page. If these tabs are not accessible to these types of log-ins, it would be beneficial to the user if they were not actually visible to them.

In regards to robustness and compatibility, the ePET software worked without issue.

#### **10.7.5.2 Test documentation**

There was no specific testing documentation evident, although it is clear that testing has been performed on the ePET software.

#### **10.7.5.3 Comments**

The ePET software was found to be very compatible and robust, with few errors.

#### **10.7.5.4 Accessibility**

ePET is a web-based tool that uses standard HTML coding that is suitable for accessing using a screen reader. There are no Flash or other elements within it that could potentially cause accessibility problems.

The links are well labelled so that they can be understood when taken out of context of the web page and displayed in the JAWS Links List window.

However, there is inconsistency with the labelling of form field elements. The majority are well labelled but there are some, for example editing the SWOT self-analysis tool, that have fields listed as 'unnamed' in the JAWS Form field list window. This can make it difficult for a JAWS user to correctly identify the fields and enter the correct data.

In the bottom banner the link text 'Created by the FMCC' is actually split into two links that both have the same URL. This causes JAWS to read out the two links separately, first 'Created by the' and then 'FMCC'. This results in the meaning of the link being more difficult to understand.

In the administrator log-in some of the pages contain large tables defining the properties of the system that can be edited. Each table row has an 'edit' link in the right-hand column and these are all listed as 'edit' in the JAWS Links List window so a JAWS user cannot differentiate between the links. This could potentially make it difficult for a JAWS user to administer the system.

## **10.8 GroupLog**

### **10.8.1 Aims and objectives**

*The teaching of large cohorts of students can lead to reduced opportunities for interaction and feedback, especially with regards to group work. We have been influenced in our thinking by the way weblog engines make publishing, commenting, sequencing, archiving, sharing and dissemination of information an easy process for users. Most weblog engines, however, assume one author. We have already developed and are using a prototype of a custom solution we called GroupLog at the University of Bath which can support the contributions of many authors whilst enabling a teacher or tutor to set parameters for when contributions could be submitted and when students' responses would be published for viewing by a full cohort of students. In the context of GroupLog the conventional weblog article or story becomes either an activity defined by a teacher or a response to the tutor-defined activity by a group of students. The JISC funded phase of GroupLog aims to further develop GroupLog and offer it to the community on an open-source basis as well as report on the feasibility of developing GroupLog as a portal plug-in (or portlet).*

Background: they found Blackboard functionality limited – it does not facilitate group work. They have used open source and Open Standards from the beginning (initial prototype was built before JISC funding). A tutor assigns an activity to a group and that group decides its response. They can paste in plain text – there is no concept of individual contributions to the group answer: it is the group's responsibility. They get to see other groups' responses – this seems to be a very important feature to students. They are using IMS Enterprise specification (re: people and group management, creating groups, and allocating people to groups). The SWEET system <http://www.brock.ac.uk/sweet/> from Brockenhurst College is being used to supply this. They think they are the first

outside Brockenhurst College to use this and, although it is very valuable, getting to grips with it has caused many delays. It needs MS-server to run it (or Java, but they don't have Java skills). Theoretically, though, what they have designed will be able to consume a Java-based web services implementation of IMS Enterprise.

## **10.8.2 Brief comments on software and code delivered**

The GroupLog software is certainly fit for purpose, meeting its original objectives.

Most software issues concerned functionality rather than problems with robustness or use of differing operating systems. The software is also considered accessible; however, a number of minor changes would allow it to meet more of the WAI priority guidelines criteria.

The code was considered excellent in all areas apart from the noting of the date. This project's software is considered particularly maintainable and extensible.

## **10.8.3 Summary of usability testing**

The interface is basically usable, but needs some work before it is suitable for unsupported use. The project team are aware that there are problems and have already proposed new designs that would improve the usability of the system.

## **10.8.4 Summary of code checking**

Excellent commenting, although date created was missing from all files. Generally excellent indentation and naming conventions. Formatting is good and the code is generally consistent, logically broken down, and uses OOP where possible. Some minor suggestions for improvement, including one security suggestion relating to 'include' files.

## **10.8.5 Summary of software testing**

### **10.8.5.1 Robustness and compatibility**

Because the program did not make use of applications such as Flash, it did not create issues with screen readers or compatibility issues within a variety of operating systems.

Bugs found were mainly concerned with functionality rather than issues in regard to compatibility. The GroupLog application is available to users of a variety of operating systems in this respect.

The GroupLog application would be considered robust but not free of errors.

### **10.8.5.2 Test documentation**

While no testing documentation was provided, there were user instructions as well as presentations on the goals of the program. The user instructions were useful and informative, using visual guidance as well as text-based instruction.

### **10.8.5.3 Comments**

Several problems were discovered:

- There are formatting options when entering the activities. However, the formatting is not saved correctly when the activity is saved. Also, when creating an activity, the dates entered at the bottom were ignored and the start and end dates were reset to 01/01/1970. The user has to alter the dates in My Admin/Manage Activities/Schedule Activities before the correct dates are saved.
- The navigation was not intuitive. The user often found difficulty in remembering how to return to a specific page.
- When the user used the [Back] and [Forward] browser navigation buttons it sometimes caused form data to be entered more than once.
- When creating an activity it is automatically assigned to the tutor's group. Therefore it is possible to schedule an activity without it actually being assigned to any student groups. The user should not be able to schedule an activity unless it has been assigned to at least one student group.

From the maintainable and extensible perspective in regards to testing, GroupLog is adequate as it's available to a wide range of users and documentation on its use is clear. However, there was no evidence of use of work packages or any other system of version control.

#### **10.8.5.4 Accessibility**

The website was generally easy to navigate using JAWS as it didn't make use of plug-ins such as Flash. However, the options to format the text when creating activities could not be tabbed to, so are not accessible to a keyboard-only user.

On the home page not all of the dropdown boxes have descriptions, which makes it too difficult for a JAWS user to navigate.

Many of the text links are surrounded by square brackets. This causes extra unnecessary text to be read out by JAWS.

## **10.9 Horus**

### **10.9.1 Aims and objectives**

- Develop and make available reusable, open-source application services that make learners and teachers more aware of curriculum goals, and gather, analyse, and disseminate feedback relating to work-based placements and other learner-centred activities.
- Develop standard service interfaces for these innovative application services.
- Integrate the developed services in the JISC technical framework.
- Develop a set of use cases in learner-centred education, to help prospective users apply the services.
- Demonstrate how these services support two e-learning scenarios.
- Disseminate the product to the JISC community by expanding the existing community of developers and users.

The primary deliverable on this project is a set of open-source services that facilitate learner-centred, work-based education. This set of services will then form part of two educational demonstrators.

### **10.9.2 Brief comments on software and code delivered**

The Horus project provided all code and software needed for evaluation.

The software has met its objectives; however, there were a number of compatibility issues.

The software is accessible up to a point, but would require a number of changes to make it fully accessible.

The Horus project code is fit for purpose but is let down in a number of key areas.

### **10.9.3 Summary of usability testing**

Their main deliverables are services and their end users are ISIS developers, so a usability walkthrough was not appropriate.

### **10.9.4 Summary of code checking**

In general, the commenting of the code is of a high standard. All fields and methods are well commented and class documentation is provided. Indentation of the code is logical and consistent throughout. Method and field names are generally consistent in terms of their case. White space is used effectively to separate logical blocks of code within methods. Various minor issues and improvements were identified.

### **10.9.5 Summary of software testing**

#### **10.9.5.1 Robustness and compatibility**

HORUS is a web-based, e-learning feedback tool which for the purposes of functionality testing is being tested on IE 6.0 in a Windows XP environment. Firstly, the log-in was swift and simple. The supplied log-in and password brings the user straight to the main screen. All tabs and navigation buttons function as expected and there were no significant loading delays when moving between sections. The diary and calendar functioned exactly as expected, allowing the user to input feedback about previously attended learning modules, and to learn of upcoming modules. The portfolio section initially caused problems as the test system did not have the required Adobe SVG viewer in order to display the active graphical output of the page. Once this was rectified, the portfolio could be seen to actively update the user feedback information; in addition the SVG functioned interactively as expected.

The only real issue to come up whilst testing through the matrix was that Firefox 1.0 does not support the new SVG plug-in and therefore will not display the active graphical output. The same was true for the Firefox browser on the Mac (although Safari functioned fine) and for Konqueror on Redhat Linux.

Due to the SVG plug-in, the Horus project is not particularly compatible, although this is due to a technology plug-in that has not yet been adopted by two major browsers. However, it is likely that within the near future these browsers will allow for this plug-in.

#### **10.9.5.2 Test documentation**

Horus provided excellent testing documentation in the form of a test plan.

It is very well thought out and follows many software QA industry best practice strategies; no doubt this is one reason why the Horus project produced a robust program.

They also provided installation and integration documents, all of which are of great use to any potential users or developers looking to maintain or extend this project.

#### **10.9.5.3 Comments**

The Horus project has used a new plug-in, which is not available for all browsers; this will require users to install specific browsers. This does limit the compatibility; although it is likely that in the future the plug-in will be developed for other browsers.

There were also shortcomings with the accessibility, which could possibly be fixed by making the Horus project meet with many of the WAI priority guidelines.

However, the documentation produced is of a particularly high standard. The test plan produced is on a par with any developed for the software QA industry.

#### **10.9.5.4 Accessibility**

Horus is a web-based project, which makes it suitable for screen reader access, and JAWS navigated the pages in a logical manner. However, as JAWS reads from left to right and top to bottom the content of the Calendar/Diary is not read in a logical order. All of the entries for a given time are read out for the whole week rather than reading each day at a time. This could make it extremely difficult for a visually impaired user to understand.

The assessment feedback pages are read by JAWS without difficulty. However, due to the extensive use of radio buttons, JAWS has a lot of descriptive text to read out, so the meaning of the page could easily get lost.

There are a lot of examples of links having the same names, for example the 'view' and '(?)' links. When they are taken out of the context of the page and listed in the JAWS Links List window the user is not able to distinguish between them.

## **10.10 Interactive Logbook**

### **10.10.1 Aims and objectives**

Interactive Logbook (IL) will support learning activities such as personal development planning, portfolio recording, assessment, and the somewhat different timetabling requirements of learners. It will provide a common interface through which learners may access the full range of client and server side tools available to them.

Reimplementation of the existing application:

- A version of the client for the J2ME platform together with the same plug-ins for connection to institutional web services as above.
- Improvements to the Windows client, to allow for an open plug-in architecture. This client will be built on the .NET architecture, and will also be tested on the Mono platform.
- Tools currently unique to the IL will be reconfigured for the plug-in architecture, allowing them to be changed for better/more appropriate ones as they become available.

New tools:

- A tool enabling learners to create a profile containing their PDP and portfolio.
- A set of plug-ins that provide direct access from the IL to various institutional learning functions such as WebCT Vista, Microsoft SharePoint, and SCT Luminis Portal (built on uPortal).

- Where web services do not exist (and are not in development) for key learning functions of the above institutional systems, where practicable, web services will be built on those systems' APIs.

### **10.10.2 Brief comments on software and code delivered**

What has been developed is basically a shell and an architecture. The applications made available for test were:

- Windows client with core plug-ins
- Mobile client with core plug-ins.

Interactive Logbook provided all their software and code for the entire project.

Despite a few minor issues, everything was evaluated and the project has met its objectives. The team were particularly helpful in aiding the testing process while allowing the evaluation to remain objective.

Although there were some minor errors with the code, on the whole it is considered excellent and highly maintainable and extensible.

### **10.10.3 Summary of usability testing**

We evaluated the Interactive Logbook application for Windows. Following the advice of the project team, we used the user guide to direct our testing. We tested from the point of view of a learner setting personal objectives and evaluating their actions against them. The application can be integrated with Open Office or Microsoft Office 2003. Due to time limitations we only tested with Microsoft Office 2003.

The software was slightly difficult to use, partly because of inconsistencies in the interface. It is currently only suited to computer-literate or highly motivated users. However, the software we tested was released in mid-March, so improvements may have been made since.

### **10.10.4 Summary of code checking**

The commenting at both code line level and function descriptor level is generally of a high standard. The code is well written and easy to read, the style is consistent throughout, and good naming conventions have been applied to class, method, and variable names. The project is broken down into logical divisions, with a separate project generated for each of the individual components and a single application that pulls all the components together. Some hard-coded file paths were found, and in addition the application appeared to be reading in hard-coded data files from the root of C drive. It is never good practice to have hard-coded paths, and in particular using the root is bad practice. It would be much better to have these set via an installer and to read these in from the registry. The code appeared to be written in a way that would lend itself to maintainability.

NB. The code we examined made much use of a third-party control to give the menus the look and feel of XP. The project team now report that this control has been removed and replaced with one created by the team, which will impose no restrictions on subsequent use or development.

### **10.10.5 Summary of software testing**

#### ***10.10.5.1 Robustness and compatibility***

The Interactive Logbook application was found to be robust, performing well on the platform on which it is designed to be used and with the two applications with which it was designed to be used.

Minimal issues were found with this program, despite being tested on a number of very different hardware platforms, including desktop PCs, notebooks, Tablet PCs, and mobile phones.

The phone compatibility did, however, prove to be an issue, as the installation on each mobile phone model/type is different and is certainly above the skill of an average mobile phone user. For each phone model, specific installation instructions would be required.

The complexity of the installation can be put down to variations in mobile phone operating systems and how easy they make it to upload and run third-party applications. Instructions on how to do this were not present.

#### ***10.10.5.2 Test documentation***

Project documentation was well kept and defined as work packages, hosted on the project website. Because of this, version control in documents, code, and program were good.

The Interactive Logbook project has provided enough information and documentation on a well-maintained project website for a third party to pick up the project easily and continue their work.

The team building the application were also very forthcoming with information.

#### **10.10.5.3 Comments**

The Interactive Logbook has a platform requirement: Windows XP with Office 2003 or Open Office. This could be considered to be a maintainability and extensibility issue because of the restriction to a specific x86 based platform.

From the maintainable and extensible point of view, Interactive Logbook is an excellent example of a well-documented, well-planned, and well-executed project.

#### **10.10.5.4 Accessibility**

Basic accessibility testing was carried out; however, this application has several issues with screen readers. We do feel, though, that with minimal adjustment it could attain a higher level of accessibility in regards to disabled access.

## **10.11 JSmirk and Smirkboard**

### **10.11.1 Aims and objectives**

JSmirk is the tool for creating presentations with easy to add audio (e.g. lecture slides with lecture itself). Smirkboard is the tool for creating an archive of presentations with comments and contributions from others (e.g. a tutor might deliver a presentation with audio and set an assignment to a group; the assignment group might add their own slides to the presentation to answer the assignment, and another group might be allowed to comment but not to add slides. The tutor might comment back on the additions.)

JSmirk aims to be a Java-based authoring tool for creating slideshow multimedia. The user will be able to draw graphic elements on a slide by slide basis, much as they would in PowerPoint, but also have the ability to add audio to the slides. Eventually the slideshow can be exported as an XHTML+SMIL or SMIL multimedia presentation or as an accessibility-guidelines-compliant HTML page. Such outputs will be bundled in SCORM packages. Smirkboard will be a server environment to which such presentations can be uploaded. However, it will also allow viewers of presentations to post dynamic comments over those presentations, which will reappear during subsequent viewings of those presentations. This attempts to combine the logic of the discussion board with that of the web presentation. Slides will appear with the comments that were provoked by them alongside. When the slideshow moves on to the next slide, those comments will too disappear and new ones pertaining to the subsequent slide will appear in their place. These learning activities will also be exportable as a SCORM package for offline perusal and evaluation.

### **10.11.2 Brief comments on software and code delivered**

The JSmirk and Smirkboard software has met its objectives; however, the JSmirk application is currently restricted to a specific version of Windows. On other operating systems much of the functionality causes errors. This clearly will restrict its user base until the JSmirk application is made more compatible.

The only code provided was for the Smirkboard side of the project. There are areas where the code is very extensible; however, it is let down in other areas by sparse and inconsistent commenting.

### **10.11.3 Summary of usability testing**

JSmirk is reasonably easy to use, particularly for those who are already familiar with Microsoft PowerPoint. The process of uploading could be made clearer, and some additional labelling and rethinking the icons would improve the overall usability.

Smirkboard is difficult to use and would require extensive usability work before it could be used unsupported. The project team are aware that there are problems.

### **10.11.4 Summary of code checking**

Since the JSmirk code was not provided, this review refers to the code provided for the Smirkboard application. Commenting is sparse and inconsistent, making for poor extensibility. All classes reviewed showed cases of nested If statements containing no braces. While legal in Java, it is strongly recommended that all If statements and further nested conditionals use the proper Java braces style. This makes the code much more readable.

In general, variable, class, and method names were descriptive, with class names adopting the standard nomenclature. Different authors applied different naming conventions in some cases. Formatting was consistent across the files. The application appears to be broken down into logical components, which makes for good extensibility. However, extensibility is let down in other areas by styles of coding that may be fast to develop but are hard to follow for new developers who might be wanting to extend or maintain the code. In addition, combining database, business, and presentation logic in the Java source code makes it difficult to debug, amend, or extend a system. Furthermore, since most files reviewed were extremely lengthy, better use of inheritance could have been undertaken to minimise repeated methods. Most files exhibited good error handling.

## **10.11.5 Summary of software testing**

### **10.11.5.1 Robustness and compatibility**

The JSmirk application was deemed robust through testing. Bugs discovered were either functional bugs within the application or to do with compatibility. All functional bugs found would be classed as minor issues: at no time did the application hang, crash, or cause a system failure.

JSmirk does, however, have compatibility issues. It will only work correctly on Windows 2000 with the latest Java JRE version installed. On other Windows operating systems some functionality did not work or parts of the application failed to operate correctly.

The Smirkboard side of the project worked as we assume it is intended to do and we had no issues with its use.

### **10.11.5.2 Test documentation**

The project team were forthcoming with information in regard to the program's design.

The project team carried out organised testing in stages using scripts; this is consistent with an organised and planned testing regime. Because of this, the project team were able to find errors that may not have been picked up in a less strict testing process. The high quality of the testing is reflected in the application.

However, the only instructional text or images provided were contained within the Smirkboard service. While this was useful, it would be preferable to provide usage instructions in the form of a text or PDF file.

### **10.11.5.3 Comments**

The JSmirk and Smirkboard is a robust application and a fresh take on producing e-learning material; it is easy to interact with and learn from. However, we found that accessibility and compatibility were both issues that could be a barrier to some potential users.

Due to the nature of the program it could (as the project team have stated) be programmed to work on other operating systems. It would also be beneficial and relatively easy to make JSmirk more accessible to users who cannot use a mouse or trackball.

Their testing programme and clear documentation are strong points of this project.

### **10.11.5.4 Accessibility**

When the user was operating JSmirk with either test screen reader (Supernova and JAWS) it was found that some of the application's functionality could not be reproduced with the screen readers. In the program the user can either use tabbing for navigation or use assigned shortcut keys. Although shortcut keys are used, the screen reader does not read these out.

The program has no issues with screen magnification tools.

We would like to make it clear, however, that content created on JSmirk for Smirkboard is only as accessible as the person creating it allows.

## **10.12 OpenMentor**

### **10.12.1 Aims and objectives**

OpenMentor consists essentially of two major parts: (a) a learning system, which acquires feedback rules through a training set, and (b) a web interface that uses these rules to provide feedback to teachers. The OpenMentor system will be available as a toolkit which can either be a standalone service or be part of a larger service-oriented architecture such as the e-Learning Framework. It will specifically make use of the Rating/Annotation service within ELF to gather information on the level and quality of tutorial support. The main aim of the OpenMentor project is

to provide a learning support tool for teachers in further and higher education, which will help them by providing reflective comments on their assessment and feedback of student assignments and coursework.

The key objectives are:

- to convert eMentor into an open source equivalent system, OpenMentor, and to ensure it meets the standards needed for adoption by IT facilities in education;
- to validate the new OpenMentor against assessment strategies used in a range of further and higher education institutions;
- to establish an essay and report assessment mentoring tool for teaching staff in further and higher education;
- to disseminate OpenMentor through the UK education community.

### **10.12.2 Brief comments on software and code delivered**

While the OpenMentor software has broadly met its objectives, there were numerous errors with the web interface, which could be attributed to either the software or the hosting of the service.

The software is fairly robust and compatible but did have some accessibility issues which, if resolved, would widen the potential user base.

The code is generally strong, although in some areas it has poor error handling.

### **10.12.3 Summary of usability testing**

The application is very usable and well suited to its target audience.

The usability testing undertaken by the project team was of a high standard and was well documented.

### **10.12.4 Summary of code checking**

Commenting was good overall but rather sparse in some areas. Formatting was consistent, code was readable, and descriptive names and standard naming conventions were used throughout. The code structure correlated directly with the written documentation. The business logic was evident in the layout of classes and their resulting methods. Good OOP implementation appeared to have been followed, with much use being made of inheritance and abstraction, and good encapsulation.

The code was broken down into simple and mainly short classes. The only exception to this was the DBHandler class, which was lengthy and complex. This could get out of hand should the application grow significantly. In addition, some SQL queries were hard-coded into the DBHandler file, which, considering its size, would be difficult to maintain or facilitate collaborative working. Error handling was not taken care of particularly well in many files. Some exceptions were caught but no resulting error handling was undertaken. The application appeared to have Test classes written to facilitate Junit testing, which is considered worthwhile and good practice.

### **10.12.5 Summary of software testing**

#### ***10.12.5.1 Robustness and compatibility***

The website essentially worked well. However, there were instances when the user encountered HTTP Status 500 error pages, specifically when the user clicked on the 'Submit and Analyse' button to submit a document and on certain systems when trying to view a 'Simple table' page. The navigation worked well and the user was able to move around the site logically without problems.

The user is able to enter a seemingly unlimited amount of characters into each edit field, causing the entered text to be truncated if there are more characters entered than will be allowed by the rest of the program. The text entry fields could be improved if a maximum character limit was defined.

The OpenMentor application can be considered a robust piece of software, working on multiple operating systems and browsers with few errors.

#### ***10.12.5.2 Test documentation***

The OpenMentor project made good use of their project website in clearly hosting documentation. There was evidence of a clear testing strategy throughout the project, although there were no specific test scripts for individual tests other than the user evaluations.

One document that was missing, however, was User Instructions.

#### **10.12.5.3 Comments**

The project team followed many industry best practice procedures. In this respect, OpenMentor is very maintainable and extensible.

#### **10.12.5.4 Accessibility**

The website is easy to navigate with JAWS, as it is written in standard HTML code. However, the home page included forum links all labelled 'more...'. These links are difficult to understand when taken out of context and viewed in the JAWS Links List Box.

When reading tables, JAWS did not pause correctly; rather, it ran straight from one line to the next, causing the meaning of the table to become lost to a JAWS user.

There is a breadcrumb trail at the top of each page beginning 'You're here:'. However, JAWS had difficulty with that phrase and read it in a way that sounded like "You're you're colon".

There is a link at the beginning of the 'Options' table on every page that reads 'om091/dummy'. This link is not visible on the page but it must be remembered that JAWS will read and allow access to all links, whether visible or not.

## **10.13 PETAL**

### **10.13.1 Aims and objectives**

PETAL addresses the lack of open-source tools designed to build e-portfolios in the UK context. It will build on good practice and re-use existing open-source software wherever appropriate. The PETAL Project aims to produce a general e-portfolio tool for lifelong learning which will address the following needs:

- a tool that is explicitly dialogic
- a tool that might be used in any post-compulsory adult education scenario and is not biased towards undergraduate university education
- a tool that is not biased towards the US education system
- a tool that is based on open-source software available to the UK education community
- a tool that is compatible with UKLeaP and the Lifelong Learning Record.

Specifically, PETAL will:

- implement CMALT as an instantiation of a general e-portfolio tool for personal and lifelong learning based on the Open Source Portfolio Initiative (OSPI)
- pilot the general e-portfolio tool for personal and lifelong learning with a small cohort of eight learning technologists from four different post-compulsory education sectors: FE, HE, ACL, and Professional Institutes
- develop eight more use-case scenarios for the application of the e-portfolio tool
- describe the processes involved in the creation and applications of a general e-portfolio tool for personal and lifelong learning using IMS LD
- create a template for the production of further use-cases of the e-portfolio tool
- disseminate knowledge, experience, and applications of the general e-portfolio tool for personal and lifelong learning to the JISC community.

### **10.13.2 Brief comments on software and code delivered**

The PETAL software objectives have been met, although the software was not considered particularly compatible and only robust on a minority of operating systems.

The PETAL software only has minor accessibility issues. It works particularly well with screen readers.

The code provided is considered poor in many areas. Error handling is not controlled well and commenting is poor. From this perspective the maintainability and extensibility of the PETAL project is poor. However, their testing documentation was good, as was the resource sharing on their project website.

### **10.13.3 Summary of usability testing**

The application is basically usable but will feel slow and frustrating to some users. Further work on the interface is needed, but the project team are already aware that this needs to be addressed.

### **10.13.4 Summary of code checking**

Only about half the files had good commenting. Not one file tested had a class descriptor and no method comments or line comments were in existence. Standard naming conventions were used throughout the project. The formatting and consistency varied from file to file. Some files contained commented out debug information and //TODO comments, and some classes had methods commented out with no explanation. Since a UML diagram couldn't be located it was difficult to decipher the design. Error handling was not tightly managed. Some methods were long and complicated and could have been broken down into smaller components. The use of hard-coded strings was generally avoided. Overall, it is highly unlikely that a new developer would be able to pick up this project and understand it.

### **10.13.5 Summary of software testing**

#### ***10.13.5.1 Robustness and compatibility***

We tested PETAL across different platforms with various browsers. Win NT SP6 with IE SPI and Win XP Pro SP2 with IE6 SP2 are the only combinations which seem to have near full functionality. With all other platforms on the matrix, the user is unable to progress past step 2 (Choose content) of the 'Create presentation' part of the program. The user was unable to select content for this and the program won't allow the user to progress without choosing content.

Across all platforms, the 'Share - comments from others' and 'View – comments from others' sections say "Page 1 of 0".

In Win XP and Win 2000, using Firefox 1.0, when the user is in 'Set up presentation', they are unable to type anything in the 'Description' section. The Word style tools are also unavailable. Both of these aspects of functionality were available to the user in Win NT SP6 with IE SPI. On Mac14 in Safari 1.2, the user can enter text here but the word style tools are unavailable.

Functionality was very limited and incorrect when using Opera. For example, in 'Set up presentation' the user marks the [never expire] box, yet the date selector for expiry date is still visible. This shouldn't be so.

As the user progresses through this section, the area that they are in, e.g. '2.Choose content', should be highlighted, but in Opera it isn't.

In the 'Upload files' section, the [browse] button doesn't say 'Browse'.

In 'Enter/Update Data' the tabs are not visible, e.g. Education, Career etc.

There were also a number of cosmetic and easily fixable errors, such as spelling and grammatical errors.

To summarise, the PETAL software was not particularly compatible. However, on the operating systems on which it did work, it was robust.

#### ***10.13.5.2 Test documentation***

PETAL made use of their own project website to host a multitude of useful documents. Although there was no specific test plan, there was evidence of testing in the form of an issue log.

They also produced a work package outlining the project's development, which meant that it was easy to backtrack development.

#### ***10.13.5.3 Comments***

Despite PETAL having issues with compatibility, it is very accessible, which is important in the education sector.

Their issue log is also a useful resource for potential developers looking to maintain or develop the existing software.

#### **10.13.5.4 Accessibility**

The PETAL project is web based, which makes it accessible with screen readers such as JAWS. The pages read in a logical order and the tabbing works correctly. The data entry screens make extensive use of forms and the fields are generally well labelled so that the JAWS user is able to understand the meaning of the fields when they are listed in the JAWS Form Field List window.

The exception is on the 'Enter/Upload Data' page where all of the data entry links are labelled generically. Therefore the user is not able to differentiate between the different links when they are displayed in the JAWS Links List window, thus making it very difficult for a visually impaired user to correctly enter their personal data into their portfolio.

### **10.14 RAMBLE**

#### **10.14.1 Aims and objectives**

The aim of this project is broadly to enhance support for learner reflection in their PLE, their own evolving, individual learning space, in such a way that they can share and receive selected response to their reflections in the context of virtual learning environments (VLE). The project will fulfil these aims through the use of web logs (blogs) as a reflective authoring activity. The specific objectives are:

- the creation of a new blogging component that will allow web log content to be incorporated and viewed in Bodington, an open-source VLE
- the provision of guidelines on how similar components could be developed for other e-learning systems
- the provision of guidelines to set up and support the offline authoring of web log entries on a PDA, the subsequent uploading to a web log server, and the incorporation in a VLE.

The project's work is divided into two strands: the offline authoring of web log entries on a PDA and subsequent upload to a web log server; and the creation of a blogging component that allows web log content to be integrated into Bodington, an open source VLE. The second strand has seen the development of a tool enabling blog content to be presented in the wider educational context of a VLE. The tool is able to query any blog server that provides standard syndicated feeds.

The project team aimed to provide reasonable coverage of common scenarios by using two kinds of PDAs for requirements gathering - HP iPaq 1940, based on MS Windows Mobile 2003 (Pocket PC), and Palm Zire 72 running Palm OS 5.2. The blog server that was installed is Pebble, which supports XML-RPC-based communications using Blogger API and MetaWebLog API; and several standards-based feeds including RDF1.0, RSS2.0, and Atom0.3. The main development has been a new Blog Resource (tool) for the Bodington VLE.

#### **10.14.2 Brief comments on software and code delivered**

The RAMBLE project software is fit for purpose and objectives have been met. The software is accessible and generally compatible and robust.

The code was mixed in quality but considered good in many areas such as error handling and commenting.

#### **10.14.3 Summary of usability testing**

Usability testing is of limited relevance to this project. The RAMBLE project team have created a clear and usable interface which is intuitive to use, providing suitable feedback to users.

#### **10.14.4 Summary of code checking**

The standard of commenting in the RAMBLE files reviewed was quite good, although various minor improvements were identified. Readability was of a good standard and all variable, method, and class names reviewed were named consistently and sensibly. Formatting was consistent and the application appeared to be logically broken down into separate units. Error handling was good in the classes reviewed and the design of the application appeared to be rational and should lend itself well to extensibility.

Note: The source code provided appeared in a txt file, so a sample of files were removed into separate Java files for ease of reviewing.

## 10.14.5 Summary of software testing

### 10.14.5.1 Robustness and compatibility

The site seems fairly stable on the whole and most of its core functionality works as intended. The only serious bug encountered during the test occurred when the user tried to view files that they had uploaded.

Some difficulties were encountered in navigating and understanding the site's layout, especially as no instructions were provided on how to use it. The icons in the navigation bar at the top of the screen aren't self-explanatory, and the alt text that accompanies them isn't particularly helpful either. Also, the instructions that feature within the site seem quite jargon heavy, meaning that users without specialist knowledge of web logging would probably find it difficult/impossible to understand them.

The RAMBLE software is compatible with various operating systems and browsers and is robust. Most issues, apart from the problem viewing files noted above, are cosmetic and probably easily fixed.

### 10.14.5.2 Test documentation

RAMBLE provide (on their project website) brief yet very helpful installation and user guides to their software. It would be easy for a third party or potential user to pick up, deploy, and use this software.

However, there was no evidence of specific testing documentation, although there was evidence of testing with potential users.

### 10.14.5.3 Comments

The RAMBLE project team's website is very helpful, providing a variety of documents, files, and other useful information on the RAMBLE project and its software.

Previous versions of software and source code are available, as well as links to active sites where the RAMBLE software is in use.

RAMBLE is very much maintainable and extensible given the information they provide.

### 10.14.5.4 Accessibility

The RAMBLE project is an extension for the Bodington VLE and as such is web based and accessible to screen readers such as JAWS.

In the 'Blogs' section each link has a 'template\_manage' link to the right of it. However, these links are have no descriptions given to them, so are all shown as 'unnamed' in the JAWS Links List window. This would make it difficult for a JAWS user to understand the purpose of the links.

The website makes use of Forms for editing pages and uploading links and all of the form fields are well labelled.

## 10.15 Serving Maths

### 10.15.1 Aims and objectives

The project aims to develop open-source software tools to address the special requirements of mathematics in the context of e-learning and e-assessment. The focus is on making existing tools user friendly, accessible, and interoperable. Among them are:

- AiM: a self-testing and assessment system based on server-side marking using the computer algebra system Maple
- CABLE: a system similar to AiM, using open-source computer algebra systems like Maxima or Axiom
- METRIC: a self-testing and mathematical learning system based on client-side marking implemented in Java
- WaLLiS: an interactive learning environment that focuses on providing adaptive and intelligent feedback
- Moodle: a user-friendly and modular virtual learning environment.

The deliverables produced by the project will include:

- a web service taking care of the web delivery of mathematical expressions
- an assessment question type to assess higher mathematical skills, extending the IMS Question and Test Interoperability specification

- an implementation of a web service based plug-in architecture, extending existing assessment systems by new question types
- authoring tools for mathematical assessment questions
- tools for integrating CAA into weekly problem sheets.

### **10.15.2 Brief comments on software and code delivered**

The project has delivered:

- Remote Question Protocol (RQP), which makes it possible for teachers and students to access many different assessment systems through the same interface
- Math QTI – an extension of IMS QT12
- an authoring tool for Math QTI
- SMITE – a web service for translating between math mark-up languages
- alterations to existing tools to make them work with these standards.

Serving Maths provided the evaluators with all the necessary code and an example of the live software to test.

The code is considered fit for purpose but there is room for improvement in some areas such as commenting.

### **10.15.3 Summary of usability testing**

The part of the project that was available for usability testing was Writing and taking a test using the Moodle VLE. The project team have integrated several different new question types and features into Moodle, and these are available as part of a demonstration on the site.

As the interface is provided largely by Moodle, there are limits to what can be tested. Design and navigation are therefore not considered.

The extensions to Moodle produced as part of the Serving Maths project are very usable, and have a well-written user guide. Usability testing is of limited relevance to the project as the Moodle extensions only constitute part of the work.

### **10.15.4 Summary of code checking**

The MQAT module contains well-documented and structured code that looks reasonably easy to maintain. It exhibits consistent naming of classes, methods, and variables and good formatting throughout. The code is logically broken down and demonstrates good use of best practice object-oriented programming.

The Wallis-mathqti module has very little formal documentation, although the line comments are quite good. We suggest using some external files for various properties. Variable, method, and class names are quite consistent, although the formatting is not.

### **10.15.5 Summary of software testing**

#### ***10.15.5.1 Robustness and compatibility***

The teacher demo site loaded correctly on the functionality test system and the site had no obvious problems with graphics, dead links, etc., with all navigation and information present and correct. Though the design of the quiz designer was a little complex initially, the actual process of creating and inserting questions was logical and did not cause any problems. The forums and resources also worked as expected.

A problem that seemed to occur with many systems that were used for compatibility testing purposes was that the log-in page frequently failed to recognise that cookies were enabled in the browser. This could be rectified with most versions of IE; however, it was not possible to bypass this using Firefox 1.0 and hence it was impossible to test using this browser on both PC and Mac systems. Also, when testing on a Mac, some of the navigation elements worked erratically or in some cases not at all.

However, in general the software is considered robust and compatible.

#### ***10.15.5.2 Test documentation***

Although no specific testing documentation (such as a test plan or scripts) was supplied, the project team showed active participation and interest in testing.

The Serving Maths project has its own Bugzilla issue tracking software (which we consider excellent practice) and discussion forums containing a considerable amount of information. There is also an active community surrounding the project, which is very helpful to potential users and developers of open source software.

The Serving Maths project website as a whole contains a comprehensive resource bank of material on the project and its associated software, and there is clear evidence of the project team taking great care over their project.

#### **10.15.5.3 Comments**

The Serving Maths project website contains a large amount of well-ordered information which will make a vital contribution towards the maintainability and extensibility of the software.

#### **10.15.5.4 Accessibility**

The user noticed that there are a few errors with the program. For instance, when the user enters an answer into the answer field and presses continue on the screen reader, the screen reader moves to the top of the page and starts to read the whole page contents again.

As this is a web based service, it could meet many WAI priority guidelines with minor changes.

## **10.16 Shell-fish**

### **10.16.1 Aims and objectives**

The Shell-fish project aims to deliver a learning support system by providing learners with a facility to obtain feedback, a development plan, and a record of progress related to that feedback. Teachers will have a feedback management system which will allow the accessing and updating of a learner's record of progress and previous feedback, linking students to existing online learning resources and monitoring the impact of resource use on students' academic performance.

The aim is to allow tutors to receive assignments in rtf and feed back comments etc. It also allows peer to peer comments (i.e. other students). It was initially conceived to fit with the Shell project but now can stand alone as an ELF component.

### **10.16.2 Brief comments on software and code delivered**

The Shell-fish project software can be considered to have met its objectives. The software is particularly accessible.

The code for the project was provided and considered maintainable and extensible.

### **10.16.3 Summary of usability testing**

The application is currently difficult and frustrating to use. Most of the problems we encountered were related to navigation and the overly complex paths through the site that users are forced to follow.

The team have undertaken their own usability testing and produced a list of recommendations. Implementing these recommendations would improve the usability of the system.

### **10.16.4 Summary of code checking**

In general, clear descriptive commenting has been applied throughout the code. The code is not very well indented. Function and variable naming was consistent and reflected their roles adequately. With regard to readability, the code was of an acceptable standard. Extensibility and maintainability are good. The navigation is driven by the database, which will make it considerably easier to update the application at a later date. Nothing was found that would indicate that there would be any major issues in maintaining the application. All application variables such as database name and server are located in a central place, making them easy to change later. As far as could be ascertained, there were no major problems that would affect efficiency bar a few minor issues.

### **10.16.5 Summary of software testing**

#### **10.16.5.1 Robustness and compatibility**

There were no particular issues with the Shell-fish project due to its simplistic approach. Shell-fish can be considered as robust and compatible.

The bugs that were found were functionality based and not caused by OS conflicts.

#### **10.16.5.2 Test documentation**

The Shell-fish project made good use of documentation and work packages. It had previously conducted constructive user testing, which is evident in the quality of the program.

#### **10.16.5.3 Comments**

The issues found during testing were minimal; this would indicate a previously well-tested program. The simple interface is also very accessible. Usability is covered in the usability review.

#### **10.16.5.4 Accessibility**

The Shell-fish site uses standard HTML code, which does not cause any problems for JAWS or other screen readers. All of the link tabs are in a logical order and appear correctly in the JAWS Links List Box. Form fields are well labelled and can be understood when taken out of context with the rest of the page, i.e.: when viewed in the JAWS Forms Entry Box.

The Shell-fish project, therefore, is a particularly accessible application.

### **10.17 SPWS (Skills Profiling Web Service)**

#### **10.17.1 Aims and objectives**

*To extend the Personal Development Planning (PDP) web service currently being developed by the WS4RL[1] project and, creating a portable skills framework, add a sophisticated skills reflection, profiling and guidance service. This service will be integrated into the Bodington VLE and tools developed to help teachers deploy and manage the service. The work will be exemplified in a medical context.*

*We will provide an example of a skills framework defined in terms of IMS specifications. The framework will be shown to be useable by populating it with several 'generic' key skills (e.g., IT, Communications etc). In addition we aim to demonstrate the framework in use by pitching it in a medical context. We will show how new (medical) skills can be added to the framework and will provide instructions on how to do so. A PDP web service will be developed; this will offer a skills profiling service. In order to demonstrate the use of this service we will provide an exemplar application within the Bodington VLE. Bodington's logbook facility will be enhanced to allow tutors to deploy the web service via a point and click interface. The service will also offer a search facility whereby a learner's competency map is used as the basis for a repository search. The overall aim of the project is to demonstrate how a user agent (Bodington VLE) can be integrated with a service from the Learning Domain. We will also show how previously developed services can be used as building blocks for new services.*

#### **10.17.2 Brief comments on software and code delivered**

Parts of the SPWS project can be considered fit for purpose, although parts of the project could not be tested. As a whole the SPWS software is fairly robust, though it was not without issue.

The code was provided for both areas of the software and was mixed in its maintainability and extensibility.

#### **10.17.3 Summary of usability testing**

The project team is currently moving the functionality to another system for presentation, so usability testing was not relevant.

#### **10.17.4 Summary of code checking**

In LUSID, the classes reviewed were fairly succinct and perfectly manageable and maintainable. Variable, class, and method naming was consistent throughout and there were no problems with formatting and consistency.

In Bodington II, some refactoring could be required on the larger classes to aid future maintenance. Some of the classes are very well commented. There were no problems with formatting and consistency and code was logically broken down into components. Overall, these files had the feeling of finished components that are well integrated into a larger system.

## **10.17.5 Summary of software testing**

### **10.17.5.1 Robustness and compatibility**

The layout remained generally uniform across IE- and Netscape-based browsers, apart from the bug that was reported concerning Firefox form fields. The development server did not perform well, producing blank pages on a regular basis.

As a whole the SPWS software is fairly robust, but it was not without issue.

### **10.17.5.2 Test documentation**

The SPWS project provided various testing documentation and other information regarding testing. Although there was not a specific and detailed test plan, it was clear from the robustness of the project that testing had taken place.

There was documentation regarding the use and installation of the project's software; however, it was not all in one specific place. It would be advantageous to any potential developers to have all the relevant documents held in one location and clearly labelled.

### **10.17.5.3 Comments**

The SPWS project worked well, although it is let down by its scattered documentation. Clearer and more ordered information would be easier to maintain and would extend the scope of the software.

### **10.17.5.4 Accessibility**

SPWS is a web-based project, which makes it suitable for screen reader access, and JAWS navigated the pages in a logical manner.

Although the pages all have the same navigational layout from a visual perspective, the pages that contain form fields have the [Home], [Recording] etc. buttons as form buttons rather than links. Therefore, when a JAWS user attempts to navigate the site they find that the navigation buttons [Home], [Recording] etc. are sometimes listed in the JAWS Links List window and other times are listed in the JAWS Forms List window.

The majority of links and buttons are well labelled, enabling the JAWS user to understand them when they are taken out of the context of the page and displayed in the Links List window. However, the form fields are not labelled correctly. Most are listed as 'unnamed' and none of the dropdown boxes for date entry are labelled. This makes it very difficult for a JAWS user to successfully fill in the data required when using the Form List window.

## **10.18 TIP**

### **10.18.1 Aims and objectives**

The project will develop a single sign-on environment between the Bodington open-source VLE, the Learning Activity Management System (LAMS) and the assessment software developed by the JISC TOIA project. This will allow users to work with the three software systems without needing to sign in, and is an important first step towards working with disparate software systems as an integrated solution. The project will also develop web services that allow the three software systems to provide and consume each other's services. In this way LAMS authors will be able to embed assessments within their activity sequences, for instance.

The main goal of the project is to build appropriate integration between the LAMS, TOIA, and Bodington software systems. The WebAuth single sign-on system will be used to provide Authentication and Authorisation services, and web services will be developed using the WSDL interoperability specification. Alongside the technical work, Oxford's medical faculty will coordinate the development of resources for learning statistics. LAMS sequences and TOIA assessments will be embedded within the Bodington VLE.

### **10.18.2 Brief comments on software and code delivered**

The TIP project provided all the information that was requested to test their software, although it is not specifically clear that their objectives have been met. This seems partly to do with factors outside the project's control. The development of the web services was testable.

The TIP project could provide us with all the software from the project, but not a testable installation; although the site documentation was very helpful, the code commenting was poor and inconsistent.

### **10.18.3 Summary of usability testing**

The project is an integration of existing tools, so a usability walkthrough was not appropriate.

### **10.18.4 Summary of code checking**

Commenting is generally poor and inconsistent. Indentation, formatting, and consistency are average, while naming conventions are excellent. Code is logically broken down and structured. On the whole, the code is well written but needs work on tidying up to make it extensible and maintainable for future coders. Site documentation was particularly helpful.

### **10.18.5 Summary of software testing**

It was not possible to test the TIP software. There is not currently an accessible installation. Further details may be obtained from John Harris at Epic plc.

## **10.19 TRICS**

### **10.19.1 Aims and objectives**

This existing software provides students with a virtual organisation (an interactive case study) that they can investigate via a web interface. It allows them to inspect company documents and interview staff in order to find out how the organisation operates. It has been trialled with both undergraduate and postgraduate students.

This project will provide an open-source version and extend and improve this open-source software. Specifically it will:

- create a new case that can be used in non IT-based disciplines
- improve the functionality and usability of the software, both for students who are learning and for tutors who are authoring new cases
- provide a set of standalone interactive tutorials that can be integrated with the cases in order to support students' investigations
- develop a new architecture for the software based on ELF, in order that the software can support:
  - greater re-usability of data describing organisations, e.g. XML describing the structure of an organisation, IMS QTI to define the interactions
  - easier distribution to, and use by, other educational institutions using XML, web services, and some form of content packaging, such as IMS CP.

### **10.19.2 Brief comments on software and code delivered**

The deliverables we have seen are:

- new features for the existing application (case studies) and new bolt-on tutorials for different modes of analysis about an organisation (for example, the SWOT tutorial)
- Rewriting the application to comply with open-source guidelines and conform with Open Standards
- SWOT and decision-making tutorials.

The TRICS project provided all the required information very early on in the process.

The software was of a high quality, although it was let down by a number of very small but noticeable issues. Although not all the software is complete, it seems that their objectives are being met.

The code provided was of good quality, with only a limited number of issues identified by us.

TRICS provided clear and concise documentation, which helped the process of code review and testing.

It is not completely clear that the major future development effort of the team will go into the open-source version of the software, as they are currently still using the proprietary standards and platform dependent software for their live applications.

### **10.19.3 Summary of usability testing**

Three pieces of software have been developed as part of the TRICS project: extensions to the non-open version of the Case Tool, the SWOT and Decision Making tools, and the open (uPortal) version of the Case Tool. We evaluated all three applications.

The original (non-open) version of the Case Tool is easy to use, though there are some issues that should be addressed particularly within the admin interface.

The new (uPortal) version of the Case Tool has a less graphical interface, but does offer some advantages over the original version. The design and navigation are cleaner, simpler, and more consistent than in the original. The fact that the graphical aspect is lost in favour of an expandable menu on the left-hand side makes it much faster to locate and explore the information available.

The SWOT and Decision Making tools are both basically usable, but further work could make them easier to use and make their benefits more apparent.

### **10.19.4 Summary of code checking**

Overall commenting was good, but some files did not have method commenting, which is very important. All variable, method, and class names were intuitive, sensible, and consistent. At times the class descriptors were quite short, but they were mostly clear and concise. Formatting was clear and highly consistent, code was well indented and organised throughout and was extremely readable with consistent style. This was due to the fact that the same developer was responsible for most of the code; however, this has its downside, for example, all files were missing copyright information and date created information. Files from both the portal and the engine modules were examined. They both appeared to be broken down into logical components. Some hard-coded constants were used for SQL queries, which is not good practice. Good OOP design was used in the project, and classes and interfaces are well thought out.

### **10.19.5 Summary of software testing**

#### ***10.19.5.1 Robustness and compatibility***

Issues discovered during testing were mainly cosmetic. These include spelling mistakes and formatting issues.

There were instances of missing content and database errors, which have little to do with compatibility or robustness and are most likely easily fixed. We feel that a clear testing script focusing on different potential issues could have revealed these issues.

Although there were a number of bugs discovered, we feel this does not detract from three pieces of well-built software.

#### ***10.19.5.2 Test documentation***

The TRICS project provided documentation on the project such as user cases, overviews, and descriptions of the software. They also provided an issue log, which shows testing had been done by the project team; however, it was not clear if there was a stated or formal testing methodology behind this.

Information was provided in a usable and understandable manner, which is important if the project is to be maintained and extended by another developer.

#### ***10.19.5.3 Comments***

The TRICS project team were quick to provide information and assistance. The three different parts to the TRICS project were also externally hosted, which further aided our assessment. To summarise, the project team were very helpful and the project was well documented and easy to understand.

#### ***10.19.5.4 Accessibility***

The TRICS Extensions to Pre-Existing Case Tool is written in Flash. There is no alternative non-Flash/text only version for screen reader usage. As this is the case, there needs to be descriptive text for JAWS to read out to the user. It should also be noted that Supernova does not understand Flash. Therefore, a Supernova user would have been presented the equivalent of an empty page.

The buttons in the navigation frame were read by JAWS but could not be tabbed to and, as they were programmed in Flash, did not appear in the JAWS Links List Box. This was of most concern when the user arrived at a Department Room, as these links are the only way to navigate back to the Lobby, so a keyboard only user was unable to proceed further.

The tabbing order in the lobby was incorrect, as the exit button is tabbed to before the 'minutes' button.

The buttons on the Select Department page are all labelled as 'gif/arrow'. Alt-text should only be given to graphics if it correctly describes the content and enhances the JAWS user's experience of the website.

The SWOT/Brainstorm pages had areas that were ignored by JAWS.

## **10.20 VLMA (Virtual Lightbox for Museums and Archives)**

### **10.20.1 Aims and objectives**

VLMA is developing both a user tool (applet) and a "resource discovery server" on the collection/server side. The applet is well documented. There should be also a white paper on RDF on SourceForge - reviewer's note – we could not find this white paper.

The idea behind the "resource discovery server" is that collection servers such as Ure will "know" about other servers and either send you to browse or search across multiple holdings. They have created RDF to translate holdings information and "suck" this in – they call this process "resyndication". They generate triples from multiple diverse databases; the basic fields are Image, Metadata, and Annotation (any of these could be empty). The server will deliver web services specific to VLMA, and the web service allows the user to save results as xml or Open Office. End user testing has taken place with students.

The aim of the project is to investigate and create structures and methods to enable teachers and learners to compare, integrate, and export discrete assemblages of images and text from online resources such as those provided through museum databases.

Objectives are:

- to adapt an existing open-source Java applet/application - Virtual Lightbox- to be a tool through which learners and teachers may collect their 'examples' from one or more databases; organise them as desired, with the opportunity to add further content; export into the desired format (e.g., email, printout, Word, PPT, or Blackboard)
- to implement a modular extensible resource-discovery server, which will expose services in a modular fashion (on the model of the Jabber server), enabling museums to use plug-in mapping interfaces to expose their data; develop and document an RDF vocabulary for museum website resources; make use of current open standards (OAI as well as RDF) to make collection metadata available (thus ensuring that the server-client relationship is not tied to any one particular software implementation).

### **10.20.2 Brief comments on software and code delivered**

The VLMA project provided all information needed to test their software and review their code.

The software has met its objectives and the code is excellent, satisfying all our criteria. Their objectives have clearly been met, although work on the interface is required. There are also a number of accessibility issues.

The Virtual Lightbox is considered to be very maintainable and extensible and the software robust and compatible.

VLMA code and software is a good example of industry best practice, although the interface requires work to make it usable.

### **10.20.3 Summary of usability testing**

The Virtual Lightbox is an interesting tool, but extensive work is needed on the user interface before it is suitable for use even by more experienced and motivated users such as teachers.

### **10.20.4 Summary of code checking**

Excellent commenting throughout. All files were found to contain good indentation. Naming conventions of classes, functions, and variables is consistent. Formatting was consistent across the project. All of the class files examined were found to be small and concise. Functionality at class level has been broken down into small manageable blocks. The code makes extensive use of design patterns and loose coupling helps ensure high maintainability and extensibility. UML Diagrams coupled with good commenting should ease the task of maintaining the code. Use of good design methodologies has decreased the complexity, and this substantially increases the ease with which it may be maintained and extended in the future.

## **10.20.5 Summary of software testing**

### **10.20.5.1 Robustness and compatibility**

The Virtual Lightbox program worked across various x86 and Macintosh platforms without serious issue. It remained a stable program throughout its use, with a number of minor issues being discovered.

The program was felt to be slow, although this may be down to poor server performance outside of the control of the project and should not reflect on the quality of the program itself.

The program interface did prove problematic with all those who tested it, but this is covered within the usability study.

### **10.20.5.2 Test documentation**

The VLMA project used the SourceForge repository to host documents. They had clearly done testing and had shown evidence of this. However, there was no evidence of a specific test plan or test scripts.

A user document was also supplied, which makes use of text and graphics to aid the user in the use of the Virtual Lightbox application.

### **10.20.5.3 Comments**

The VLMA program worked to its design, as far as we could ascertain. The main issues we discovered in testing were in regard to its interface.

### **10.20.5.4 Accessibility**

The accessibility of Virtual Lightbox was varied; we do feel improvements with a redesigned interface would significantly affect the accessibility of the program as much as the usability. The VLMA project does not pass any WAI priority guidelines.

## **10.21 V-MAP**

### **10.21.1 Aims and objectives**

The broad aim of the project is to provide a software tool that will be more accessible to dyslexic people and better enable them to produce an e-portfolio than existing tools.

The objectives of the project are to:

- enable the learner to interact with a pre-defined institutional template for preparing e-portfolios in a format that is most accessible to the learner
- enable the learner to plan, construct, and update an e-portfolio through the use of a visual mapping interface, on their own personal desktop
- enable the learner to share, publish, and disseminate an e-portfolio through the use of a visual mapping interface, to their institutional system.

### **10.21.2 Brief comments on software and code delivered**

The project is a graphical/concept-mapping approach to building e-portfolios. The software produced includes:

- Java desktop application for creating and editing e-portfolios
- a Moodle plug-in
- APIs.

The V-MAP project provided the evaluators with their code and an example of the live system.

The software has clearly met its objective and the code is of a high quality, although there is insufficient error handling.

The code is maintainable and extensible.

The project produced very limited supporting documentation.

### **10.21.3 Summary of usability testing**

We looked at the system from the point of view of a tutor creating a template, then passing it to a student to complete.

We found the software to be reasonably easy to use. Although there are still issues to be addressed, V-MAP provides an engaging way of creating and maintaining portfolios.

Some of the more advanced presentation features may not be necessary and could be removed. More explanation of the concepts should be provided, preferably within the application.

### **10.21.4 Summary of code checking**

Most files had good commenting and were logically indented. All class, method, and variable names were named sensibly and all files were well formatted and consistent. The application appeared to be broken down into logical parts and good OOP design was implemented in this project. There was insufficient error handling evident in the classes reviewed.

### **10.21.5 Summary of software testing**

#### ***10.21.5.1 Robustness and compatibility***

The V-MAP software can be considered to be robust and compatible, working on various Windows and Macintosh operating systems. When testing for operation on the Linux operating system, we found that it failed to work on Fedora 3. This is likely to be a minor problem as the developer uses Linux as the main test bed and the project regularly run the software under Linux

#### ***10.21.5.2 Test documentation***

The V-MAP project clearly laid out a plan, which is evident on their project website. They followed best practice, using work packages and publishing information including quality plans, software specifications, and presentations on the project.

However, despite the mention of testing and user instructions within work packages, no testing documents were supplied.

#### ***10.21.5.3 Comments***

The V-MAP software had a professional look about it, and it was robust and bug free; however, lack of some supporting documentation may hinder any third party from maintaining or continuing with this project.

#### ***10.21.5.4 Accessibility***

The product is not suitable for accessing with a screen reader. However, JAWS reads out the text as it is being entered and edited by the user.

The menus can be accessed using the keyboard, as with any Windows application. However, the user found no way of tabbing between the activities once they are placed in the V-MAP screen. The icons in both the top and left-hand toolbars have alt-texts but are not accessible via the keyboard. Also there is no reference to accessibility or keyboard shortcuts (other than those shown in the menu dropdowns) in the documentation for the project.

To summarise, the program cannot be used without a mouse and is also not suitable for use by visually impaired users.

## **10.22 WCKER**

### **10.22.1 Aims and objectives**

Before Reload there were "packaging" standards and VLEs that could consume learning "packages". But it was very difficult to be sure that your package would really be transferable or acceptable to a particular VLE. With Reload you could be sure that the packages you created were standards compliant. So if they didn't work you could be surer that the VLE wasn't working, or was requiring non-standard content. WCKER aims to improve Reload in two ways:

- usability - Reload requires you to understand the concept of how the packages work if you want to use it
- quality - Reload starts with a blank screen; there is no guidance on how to structure the material. WCKER helps by giving the user guidance.

WCKER opens Reload to a different set of users. Using a WCKER wizard will allow authors themselves to use Reload (although to *create* a wizard you will still need a technical person – or at least someone who writes XML). WCKER will also restrict what can be done, thus encouraging well-structured content packages.

The WCKER will be configurable and capable of running a variety of wizards. The project will initially build a course specification wizard to be run by the WCKER.

Objectives are:

- work with the Reload team (and other partners) to create the Wizard Construction Kit Extension for Reload
- use the WCKER to build a course specification wizard
- evaluation of the WCKER with a view to further development
- evaluation of the course specification wizard with a view to adding more choice for the user.

### **10.22.2 Brief comments on software and code delivered**

The WCKER project provided the required information for the code and testing review. The project objectives have been met.

The WCKER code had naming convention inconsistencies, very little commenting, and included many 'TODOs'. The code's maintainability and extendibility is poor, which is in sharp contrast to the generally high quality of the application.

### **10.22.3 Summary of usability testing**

Limited usability testing was possible, as only a demonstration interface was available.

The demonstration interface is well designed and very usable. Assuming that the tool is implemented as planned, it should be well suited to a wide range of users.

### **10.22.4 Summary of code checking**

There are inconsistencies throughout the code, with very little commenting, although it was felt that all of the classes reviewed were in a state of transition. Of concern would be the naming convention inconsistencies. This project seems to be at an early stage but there is an impression of the system being broken down into logical components.

### **10.22.5 Summary of software testing**

#### ***10.22.5.1 Robustness and compatibility***

When the user selects WCKER from the Tools menu, the window that appears opens behind the main Reload Editor window. Therefore, it is not obvious to the user that the window has opened successfully.

When moving up and down the items using the up and down arrows, the 'Course title', 'Author', and 'Course aim' fields change to reflect the currently highlighted item. However, when using the [Previous Item] and [Next item] buttons to move up and down the list, the 'Course title', 'Author', and 'Course aim' fields are not updated.

When the user clicks on [Export] no requester is displayed, so the user is unable to choose the location of the exported file. Also, no message is generated when the export is completed. The user was able to locate the export folder by reading the output in the command line editor and loaded in the *imsmanifest.xml* file created. However, none of the user's changes had been exported and reloaded.

#### ***10.22.5.2 Test documentation***

The WCKER project supplied an excellent user guide, which would be helpful not only for any potential user but for any potential developer looking to take on the project.

Although the project made good use of user documentation and there was evidence of testing within their weekly meeting notes (held within a section of the website), there were no specific testing documents, e.g. test scripts.

### **10.22.5.3 Comments**

The WCKER project has built a very useful tool for an existing application used within the e-learning industry. The Reload tool is a favoured application in the e-learning development and testing cycle and the WCKER project's software makes this tool particularly easy to access.

They have presented the project and its documentation clearly and presented potential users with a very good set of instructions, using both text and images to do this.

Apart from missing specific testing documentation, and the code issues dealt with elsewhere, the WCKER project can be considered maintainable and extensible.

### **10.22.5.4 Accessibility**

The product is not suitable for accessing with a screen reader. However, JAWS reads out the text as it is being entered and edited by the user.

The menus can be accessed using the keyboard, as with any Windows application, as can the [Back], [Next], and [Finish] buttons. However, the other buttons do not have any keyboard shortcuts assigned to them and are not accessible using the tab key as the user is unable to tab past the 'course aim' text entry field.

To summarise, the program cannot be used without a mouse and is also not suitable for use by visually impaired users.

It must be taken into account that the application is a tool designed to be used with Reload; it is not accessible itself.

## Appendix I – Abbreviations and glossary

<b>ACL</b>	<b>Adult and Community Learning</b>
<b>API</b>	<b>Application Program Interface, a set of routines, protocols, and tools for building software applications</b>
<b>CAA</b>	<b>Computer-assisted Assessment</b>
<b>CETIS</b>	<b>Centre For Educational Technology Interoperability Standards</b>
<b>cvs</b>	<b>Concurrent Versions System, an open-source version control system</b>
<b>DeLeTools</b>	<b>Distributed e-Learning Tools</b>
<b>ELF</b>	<b>e-Learning Framework (see Appendix 4)</b>
<b>GUI</b>	<b>Graphical User Interface</b>
<b>HE</b>	<b>Higher Education</b>
<b>HEFCE</b>	<b>Higher Education Funding Council for England</b>
<b>HTML</b>	<b>HyperText Mark-up Language</b>
<b>IMS</b>	<b>A consortium, IMS develops and promotes the adoption of open technical specifications for interoperable learning technology</b>
<b>JISC</b>	<b>Joint Information Systems Committee</b>
<b>LAMS</b>	<b>Learning Activity Management System</b>
<b>LIFT</b>	<b>Tool available from <a href="http://www.usablenet.com/">http://www.usablenet.com/</a></b>
<b>OAI</b>	<b>Open Archives Initiative</b>
<b>OOP</b>	<b>Object-Oriented Programming</b>
<b>OS</b>	<b>open source</b>
<b>OSPI</b>	<b>Open Source Portfolio Initiative</b>
<b>PDP</b>	<b>personal development portfolio or personal development planning</b>
<b>PLE</b>	<b>personal learning environment</b>
<b>RDF</b>	<b>Resource Description Framework</b>
<b>QA</b>	<b>Quality Assurance</b>
<b>SourceForge</b>	<b>open source software development website</b>
<b>SQL</b>	<b>Structured Query Language. A specialized language for sending queries to databases.</b>
<b>Subversion</b>	<b>an open-source version control system</b>
<b>Trac</b>	<b>Trac is an enhanced wiki and issue tracking system for software development projects.</b>
<b>UML</b>	<b>Unified Markup Language , used to produce case, class, and sequence diagrams, and more</b>
<b>UUK</b>	<b>Universities UK</b>
<b>VLE</b>	<b>virtual learning environment</b>
<b>W3C</b>	<b>World Wide Web Consortium</b>
<b>WAI</b>	<b>Web Accessibility Initiative</b>
<b>WSDL</b>	<b>Web Services Description Language</b>

## Appendix 2 - The 22 projects and their websites

AcademicTalk	<a href="http://www.londonmet.ac.uk/ltri/research/projects/at.htm">http://www.londonmet.ac.uk/ltri/research/projects/at.htm</a>
ASAP	<a href="http://www.elframework.org/learning_domain_services/assessment/asap">http://www.elframework.org/learning_domain_services/assessment/asap</a>
Assis	<a href="http://www.hull.ac.uk/esig/assis_deliverables.html">http://www.hull.ac.uk/esig/assis_deliverables.html</a>
Manchester PLE/VLE Framework	<a href="http://www.jisc.ac.uk/deletbod3ple.html">http://www.jisc.ac.uk/deletbod3ple.html</a>
DELTA	<a href="http://www.essex.ac.uk/chimera/delta/">http://www.essex.ac.uk/chimera/delta/</a>
eLaws (e-Learning Annotation Web Service)	<a href="http://www.jisc.ac.uk/index.cfm?name=deleteaws">http://www.jisc.ac.uk/index.cfm?name=deleteaws</a>
ePET	<a href="http://www.eportfolios.ac.uk/ePET/">http://www.eportfolios.ac.uk/ePET/</a>
GroupLog	<a href="http://www.bath.ac.uk/e-learning/grouplog/jisc/jisc.html">http://www.bath.ac.uk/e-learning/grouplog/jisc/jisc.html</a>
Horus	<a href="http://www.csc.umist.ac.uk/horus/">http://www.csc.umist.ac.uk/horus/</a>
Interactive Logbook	<a href="http://portal.cetadl.bham.ac.uk/ilogbook/">http://portal.cetadl.bham.ac.uk/ilogbook/</a>
JSmirk and Smirkboard	<a href="http://www.jisc.ac.uk/index.cfm?name=deletjsmirk">http://www.jisc.ac.uk/index.cfm?name=deletjsmirk</a>
OpenMentor	<a href="http://kn.open.ac.uk/public/index.cfm?wpid=3817">http://kn.open.ac.uk/public/index.cfm?wpid=3817</a>
PETAL	<a href="http://www.brookes.ac.uk/research/odl/petal/petal_home.html">http://www.brookes.ac.uk/research/odl/petal/petal_home.html</a>
RAMBLE	<a href="http://ramble.oucs.ox.ac.uk/">http://ramble.oucs.ox.ac.uk/</a>
Serving Maths	<a href="http://mantis.york.ac.uk/moodle/course/view.php?id=2">http://mantis.york.ac.uk/moodle/course/view.php?id=2</a>
Shell-fish	<a href="http://www.shell-fish.org.uk/">http://www.shell-fish.org.uk/</a>
SPWS - Skills Profiling Web Service	<a href="http://www.elframework.org/projects/spws/view">http://www.elframework.org/projects/spws/view</a>
TIP	<a href="http://www.jisc.ac.uk/index.cfm?name=delettip">http://www.jisc.ac.uk/index.cfm?name=delettip</a>
TRICS	<a href="http://www.trics.mmu.ac.uk">http://www.trics.mmu.ac.uk</a>
VLMA Virtual Lightbox for Museums and Archives	<a href="http://www.rdg.ac.uk/ure/VLMA/">http://www.rdg.ac.uk/ure/VLMA/</a>
V-MAP	<a href="http://vmap.gold.ac.uk/">http://vmap.gold.ac.uk/</a>
WCKER	<a href="http://waterbuck.conted.ox.ac.uk/cgi-bin/trac.cgi">http://waterbuck.conted.ox.ac.uk/cgi-bin/trac.cgi</a>

Where the URL shown is on the JISC website, the project has no home page or it was unavailable when this report was written.

## Appendix 3 - Short author biographies

---

---

### **Nicky Ferguson**

#### **Managing Director, Clax Ltd**

Nicky Ferguson now works as a consultant and is managing director and co-founder of Clax Limited. He has recently written, with Neil Smith and Seb Schmoller, a widely-read report for JISC entitled Personalisation in Presentation Services. The report may be seen at <http://www.therightplace.net/jp/>.

He worked at the University of Bristol's Institute for Learning and Research Technology for ten years (leaving in 2003), initially as Fellow in Networked Information and subsequently as ILRT research director and acting director. He also directed a number of national and international research projects and web-based services. He previously worked for the UK Economic and Social Research Council (ESRC).

He has also worked successfully as an actor and trainer and has set up and subsequently sold two successful small businesses. He is currently involved with the UK's first new-build Cohousing development.

### **John Harris**

#### **Director of Education, Epic Group plc**

John joined Epic in 1996 having spent four years at KPMG, managing their Learning Technologies Group. John's core skills are instructional design, user interface design, project management and consultancy. John has designed and managed many innovative e-learning and web projects for Epic clients, including an award winning e-learning programme for PwC, the successful PRIME:Leadership programme for the country's top civil servants and the Houses of Parliament website. As a consultant, John has conducted a number of e-learning research projects for Epic and has written numerous reports, booklets and papers.

Since 2001 John has been developing Epic's education business, winning prestigious contracts for clients such as National College of School Leadership (Leadership e-learning resources), Becta (National Learning Network materials) and HEFCE ([www.hero.ac.uk](http://www.hero.ac.uk) and [www.tqi.ac.uk](http://www.tqi.ac.uk)).

### **Gill Osguthorpe**

#### **Director, Futurate Ltd**

Gill is a director of Futurate Ltd. She has 13 years post-graduate experience in the education and technology industries, having previously lectured at HE level, and worked as a consultant in e-media and e-learning and as a software developer and analyst at Fretwell-Downing Data Systems. She formed Futurate with her business partner Jonathan Grove in 2000 in response to the emerging online learning market. Gill has a deep understanding of usability and accessibility issues and e-learning standards. She has managed and evaluated the usability of many e-learning development projects and tools and recently was a member of the consultation teams that drafted BS 8419 and BS 8788 for the British Standards Institute.

### **William Pellet**

#### **Testing Project Manager, Epic Group plc**

Bill joined EpiCentre in 2000 and has considerable experience in software product testing. He has project managed testing for Becta and Nelson Thornes and worked on projects for BBC, Channel 4, UFI/LearnDirect, BAE, British Airways, B&Q, Oracle, Barclays, Virgin, Hewlett Packard and the Ministry of Defence. As well as project management, Bill provides consultancy advice and support on many technical issues, including hardware configurations, operating systems, networks, testing methods, interoperability, testing management, accessibility and Open standards compliance.

### **Suzi Wells**

#### **Consultant and Project Manager for Futurate Ltd**

Suzi's core skills include usability evaluation, and systems analysis and design. Her work at Futurate has included usability evaluation for the Natural History Museum, instructional design for the Environment Agency and project management for the Association for Learning Technology.

She has been working in web development since 1999. Before working for Futurate she was at OneWorld.net (a tech-focused charity, using the Internet to promote human rights and sustainable development), where she worked with organisations including British Council, UNESCO and Christian Aid.

### **Contact details and further information**

**Website address:** <http://www.therightplace.plus.com/xxxxxxx>

**Authors' contact:** Nicky Ferguson Email: [sqe@clax.co.uk](mailto:sqe@clax.co.uk)

## Appendix 4 - JISC's Layered Services Framework

