Delivery of QTIv2 Question Types.

Gary Wills[‡], Hugh Davis[‡], Lester Gilbert[‡], Jonathon Hare[‡], Yvonne Howard[‡], Steve Jeyes[†], David Millard[‡], and Robert Sherratt[†]

[‡]Learning Societies Lab, University of Southampton, UK.

[†]e-Services Integration, University of Hull, UK

Abstract

The QTI standard identifies sixteen different question types which may be used in on-line assessment. While some partial implementations exist, the R2Q2 project has developed a complete solution that renders and responds to all sixteen question types as specified. In addition, care has been taken in the R2Q2 project to ensure that the solution produced will allow for future changes in the specification. The paper summarises the rationale of Web services and a Service Oriented Architecture, and then demonstrates how the R2Q2 project integrates into JISC's e-Framework, and the reference model for assessment (FREMA[1]).

The design of R2Q2 is described, the focus being on lessons learnt. We describe the architecture and the rationale of the internal Web services and explain the approach taken in implementing the QTI specification, showing how the design allows for future tags to be added with the minimal of programming effort. A major objective of the design was to solve the problem of having to undertake a major redesign and reimplementation as a result of minor modifications to the specification.

In the 2006 Capital Programme from JISC, three new projects were commissioned in the area of Assessment: one for authoring of items, one for item banking, and one for a complete test engine as described in the QTI specification. The R2Q2 Web service is at the heart of all three projects and this paper will describe how the R2Q2 Web service will be used.

1. Introduction

Formative assessment aims to provide appropriate feedback to learners, helping them gauge more accurately their understanding of the material set. It is also used as a learning activity in its own right to form understanding or knowledge. It is something lecturers/teachers would love to do more of but do not have the time to develop, set, and then mark as often as they would like. A formative e-assessment system allows lecturers/teachers to develop and

---

[1] Framework Reference Model for Assessment http://www.frema.ecs.soton.ac.uk/

set the work once, allows the learner to take the formative test at a time and place of their convenience, possibly as often as they like, obtain meaningful feedback, and see how well they are progressing in their understanding of the material. McAlpine [11] also suggests that formative assessment can be used by learners to *"highlight areas of further study and hence improve future performance"*. Steve Draper [12] distinguishes different types of feedback, highlighting the issue that although a system may provide feedback, its level and quality is still down to the author.

E-learning assessment covers a broad range of activities involving the use of machines to support assessment, either directly (such as web-based assessment tools, or tutor systems) or indirectly by supporting the processes of assessment (such as quality assurance processes for examinations). It is an important and popular area within the e-learning community [6, 1, 2]. Within this broad view of e-learning assessment, the domain appears established but not mature, as traditionally there has been little agreement on standards or interoperability at the software level. Despite significant efforts by the community, many of the most popular software systems are monolithic and tightly coupled, and standards are still evolving. To address this there has been a trend towards Service-Oriented Architectures (SOA). SOAs are an attempt to modularise large complex systems in such a way that they are composed of independent software components that offer services to one another through well-defined interfaces. This supports the notion that any of the components could be 'swapped' for a better version when it becomes available.

One of the more popular standards that has emerged is Question and Test Interoperability (QTI) developed by the IMS Consortium[2]. The QTI specification describes a data model for representing questions and tests and the reporting of results, thereby allowing the exchange of data (item, test, and results) between tools (such as authoring tools, item banks, test constructional tools, learning environments, and assessment delivery systems) [10]. Wide take-up of QTI would facilitate not only the sharing of questions and tests across institutions, but would also enable investment in the development of common tools. QTI is now in its second version (QTIv2), designed for compatibility with other IMS specifications, but despite community enthusiasm there have been only a few real examples of QTIv2 being used, with no definitive reference implementation [8,9].

This paper presents the Web service R2Q2 and the Test delivery engine ASDEL. R2Q2 is a JISC funded project that brings the SOA approach and QTI standard together to develop a set of Web Services that will render and respond to questions written to the QTI standard. The paper will also report on the progress being made on the ASDEL project, again funded by JISC to develop a QTIv2 compliant test delivery engine.

---

[2] IMS QTI homepage: http://www.imsglobal.org/question/

2. Service Oriented Architectures

Service-Oriented Architectures (SOAs) enable large complex systems to be mutualised, that is composed of independent software components that operate through well-defined interfaces. A service approach is ideally suited to more loosely coupled systems, where individual parts may be developed by different people or organizations. Wilson *et al.* [7] discuss in detail the advantages of using a SOA: the ability to dynamically couple services, interoperability of services due to clearly defined standards, and as a result the ability to avoid technology 'lock-in'.

Due to the nature of the loose coupling in a SOA, applications can be developed and deployed incrementally. In addition, new features can be easily added after the system is deployed. This modularity and extensibility make SOA especially suitable as a platform for an assessment system with evolving requirements and standards. Services are also appealing in terms of their ability to be reused, as they have well-defined public interfaces.

One way to promote QTIv2 is through a reference implementation of the standard written within the service-oriented paradigm. In the UK, the Joint Information Systems Committee (JISC) is financed by all the Further and Higher Education funding councils, and is responsible for providing advice and guidance on the use of Information and Communications Technology (ICT) for learning and teaching. Part of their strategy is the development of a SOA framework for e-learning [5,7], and of reference models that describe how different areas of e-learning can be supported by the framework. JISC call this initiative simply the 'e- Framework'.

The e-Framework is based on a service-oriented factoring of a set of distributed core services [17], where flexible granular functional components expose service behaviours accessible to other applications via loosely coupled standards-based interfaces. The technology used is Web Services and the intention is to extend the SOA programming model into a vast networking platform that allows the publication, deployment, and discovery of service applications on the scale of the Internet.

For the assessment domain, the reference model is FREMA (Framework Reference Model for Assessment)[3]. The FREMA project has defined a number of high level service profiles that describe how services can work together within the assessment domain to fulfil particular use cases [4].

3. Question and Test Interoperability

The IMS QTI Specification is a standard for representing questions and tests with a binding to the eXtended Markup Langage (XML, developed by the W3C) to allow interchange. Figure 1 shows a short example of a question expressed in this format, taken from the IMS QTI examples. This example is a simple multiple choice question, illustrating the core elements: *ItemBody*

---

[3] FREMA homepage: http://www.frema.ecs.soton.ac.uk/

declares the content of the question itself, *ResponseDeclaration* declares a variable to store the student's answer, and *OutcomeVariables* declares other resulting variables, in this case a score variable to hold the value of the result.

```
<?xml version="1.0" encoding="UTF-8"?>
<assessmentItem xmlns="http://www.imsglobal.org/xsd/imsqti_v2p0"
    identifier="choice" title="Unattended Luggage"
    adaptive="false" timeDependent="false">
    <responseDeclaration identifier="RESPONSE" cardinality="single"
                         baseType="identifier">
        <correctResponse>
            <value>ChoiceA</value>
        </correctResponse>
    </responseDeclaration>
    <outcomeDeclaration identifier="SCORE" cardinality="single"
                        baseType="integer">
        <defaultValue>
            <value>0</value>
        </defaultValue>
    </outcomeDeclaration>
    <itemBody>
        <p>Examine the following sign:</p>
        <p>
           <img src="images/sign.png" alt="NEVER LEAVE LUGGAGE UNATTENDED"/>
        </p>
        <choiceInteraction responseIdentifier="RESPONSE"
                           shuffle="false" maxChoices="1">
            <prompt>What does it say?</prompt>
            <simpleChoice identifier="ChoiceA">You must stay with your
                 luggage at all times.</simpleChoice>
            <simpleChoice identifier="ChoiceB">Do not let someone else look
                 after your luggage.</simpleChoice>
            <simpleChoice identifier="ChoiceC">Remember your luggage when
                 you leave.</simpleChoice>
        </choiceInteraction>
    </itemBody>
    <responseProcessing template =
    "http://www.imsglobal.org/question/qti_v2p0/rptemplates/match_correct"/>
</assessmentItem>
```

Figure 1: Example QTIv2 question (abridged for simplicity)

In R2Q2 we focus on rendering and responding to the 16 different types of interactions described in version 2 of the QTI specification (QTIv2). These are:

1) Choice
2) Hotspot
3) Order
4) Select point
5) Associate
6) Graphic
7) Match
8) Graphic Order
9) Inline Choice
10) Graphic Associate
11) Text Entry
12) Graphic Gap Match
13) Extended Text
14) Position object
15) Hot Text
16) Slider

The list of different question types can be combined with templated question or adaptive response profiles, providing an author with numerous alternative

methods for writing questions appropriate to the needs of the students. Templated questions include variables in their item bodies that are instantiated when a question is rendered (for example, inserting different values into the text of maths problems). Adaptive questions have a branching structure, and the parts that a student sees depends on their answer to each part of the branch. In total these allow for sixty-four different possible combinations.

4. R2Q2 Design

The R2Q2 service allows a student to view a question, answer a question, and view the feedback. The R2Q2 engine (see Figure 2) is a loosely coupled architecture comprising of three interoperable services. All the interactions with and within the R2Q2 engine are managed by an internal component called the Router.

The Router is responsible for parsing and passing the various components of the item (QTIv2) to the responsible web services. It also manages the interactions of external software with the system, and it is therefore the only component that handles state. This enables the other services to be much simpler, maintaining a loosely coupled interface but without the need to exchange large amounts of XML.

The Processor service processes the user responses and generates feedback. The Processor compares the user's answer with a set of rules and generates response variables based on those rules. The Renderer service then renders the item (and any feedback) to the user given these response variables.

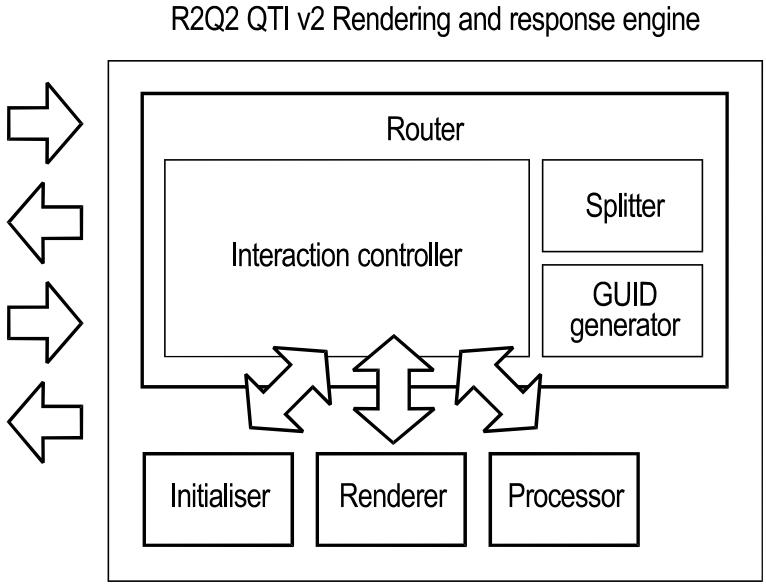

Figure 2 The R2Q2 Architecture

4.1 Integration into a Portal framework.

Figure 2 shows the core services where R2Q2 is used as a stand alone service. To ensure wide-spread take up of the Web service, R2Q2 is also designed to be dropped into applications such as a VLE, portal framework, and test engine authoring tool, amongst other applications, to achieve the aim of migrating the community to this new standard. To this end the project Web site provides documentation for installation, and a single install process.

When integrating Web services with VLEs and portal frameworks, we have found that you cannot just call a service, but code needs to be written to manage calls to and information from the Web services. The generic name for such a piece of code is an adaptor (see the EFSCE project[4]).

The R2Q2 project provides a demonstrator in the form of a Web client that uses traditional XHTML and JAVA servlets to display the questions. There are key differences to be considered between a portlet implementation of R2Q2 and a more traditional simple servlet implementation. The java PortletRequest object involves a protocol which is different from that of a HTTPServeletRequest object. The main difference is that the portlet requests contain additional information regarding the portlet window within the portal. As a result, the way the request is handled will be different, for example within the R2Q2 demo it is no longer possible to use the ServletFileUpload class as a file upload handler for the request.

There are a number of open source portal frameworks that are currently being used. They are all similar in that they are Java-based and use a Model View Controller (MVC) architecture. The MVC architecture separates the presentation code from the business logic code and is implemented using Struts for web applications. Struts provide a mechanism by which the flow of information is directed to the correct portlet. The way this is implemented means that the system can scale quite easily. Struts model the various functions of the portlet as ‘actions’. When an action URL is sent, a controller redirects the portlet to the correct JSP page which connects to the Web service.

5. ASDEL

R2Q2 successfully implemented a rendering and response engine for a single question (also termed an item), for which there are sixteen types described in the specification and implemented in R2Q2. While this is useful, it does not implement the whole of the QTI specification regarding the test process. The specification details how a test is to be presented to candidates, the order of the questions, the time allowed, etc. The typical use-case from the point of view of a learner candidate of the test process is illustrated in Figure 3.

---

[4] EFSCE project Web page http://www.efsce.ecs.soton.ac.uk/overview

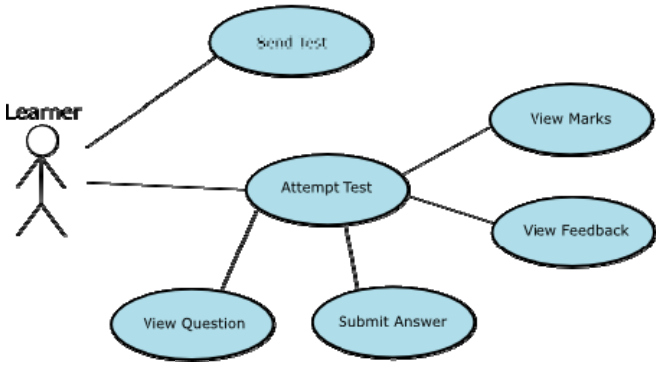

Figure 3: Use case of ASDEL from the user perspective

In the ASDEL project we aim to build an assessment delivery engine to the IMS Question and Test Interoperability version 2.1 specifications that can be deployed as a stand-alone web application or as part of a Service Oriented Architecture enabled Virtual Learning Environment or portal framework. The engine will provide for:

- Delivery of an assessment consisting of an assembly of QTI items, with the possibility that the assessment is adaptive and the ordering of questions can depend on previous responses,
- Scheduling of assessments against users and groups,
- Rendering of tests and items using a web interface,
- Marking and feedback,
- A web service API for retrieving assessment results.

Like R2Q2, the ASDEL project will use a Service Oriented Architecture (SOA). The design of the ASDEL system specifies that the major components will be created as internal Web services.

Phase 1 is the technical development of the engine in accordance with the IMS QTIv2.1 specification and in accordance with the JISC e-Framework approach of using web services in a Service Oriented Architecture (see Figure 4). The engine will take in a test as an IMS Content Package or by reference to the test XMLfile. The engine will unpack the content package and assemble the items into a directory on a local file system. The engine will import any additional material (images, videos, etc) required by the test, and it will then process the XML and deliver the test as scheduled to the candidate via a Web interface. Feedback will be given to the candidate and the marks processed in accordance with the schema sent to the engine. The results can be retrieved through the engine API. The engine will also have the additional features of being able to persist partially completed tests for future completion, and the ability to record candidate responses (in addition to results) for later review.

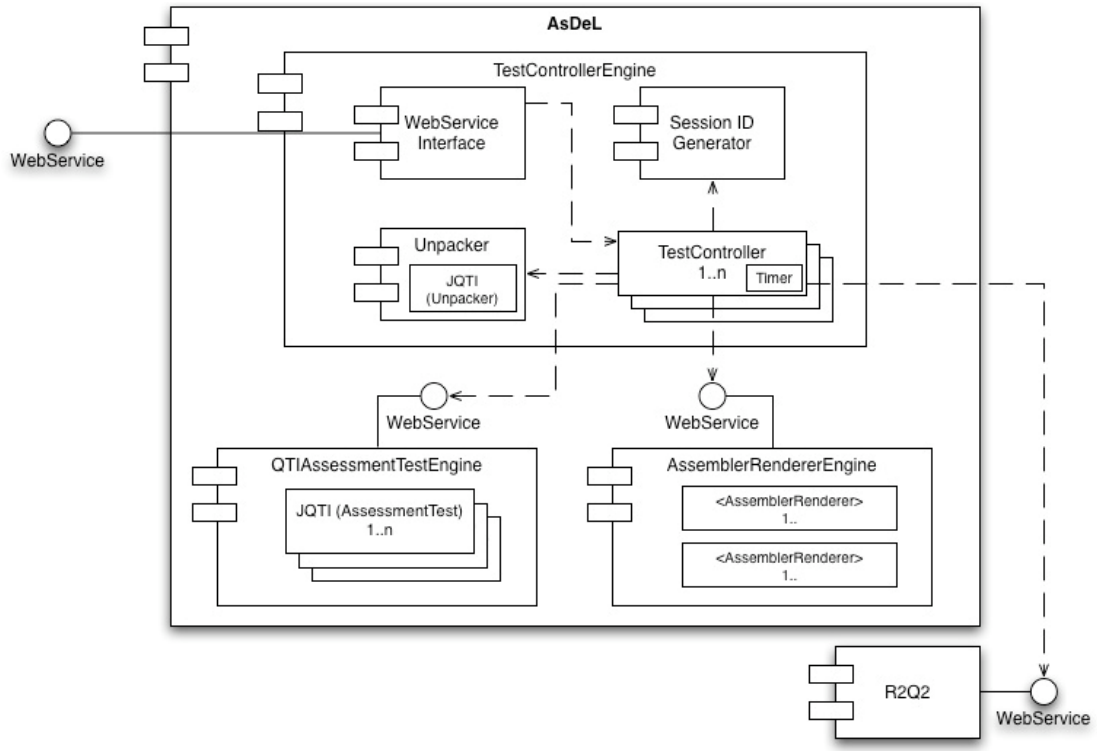

Figure 4. Architecture for the Assessment Delivery system.

The core components of the ASDEL system will be built around a Java library, which has been termed JQTI. The JQTI library will enable valid QTI assessment XML documents to be interpreted and executed. The library will also provide auxiliary services like the handling of QTI content packages and the provision of valid QTI conformance profiles and reports.

The AssemblerRenderingEngine part of the system is responsible for the assembly and rendering of output (i.e. questions and associated rubric). Initially, only an XHTML renderer will be developed; however, the design of the engine will enable different renderers to be plugged in.

Figure 5 illustrates the typical sequence of events when a user is interacting with the ASDEL system through a particular portal or VLE. Figure 6 shows the typical initialisation stages that the system goes through when a test package is presented, and Figure 7 demonstrates the typical collaborations between system parts when the a learner is undertaking a test.

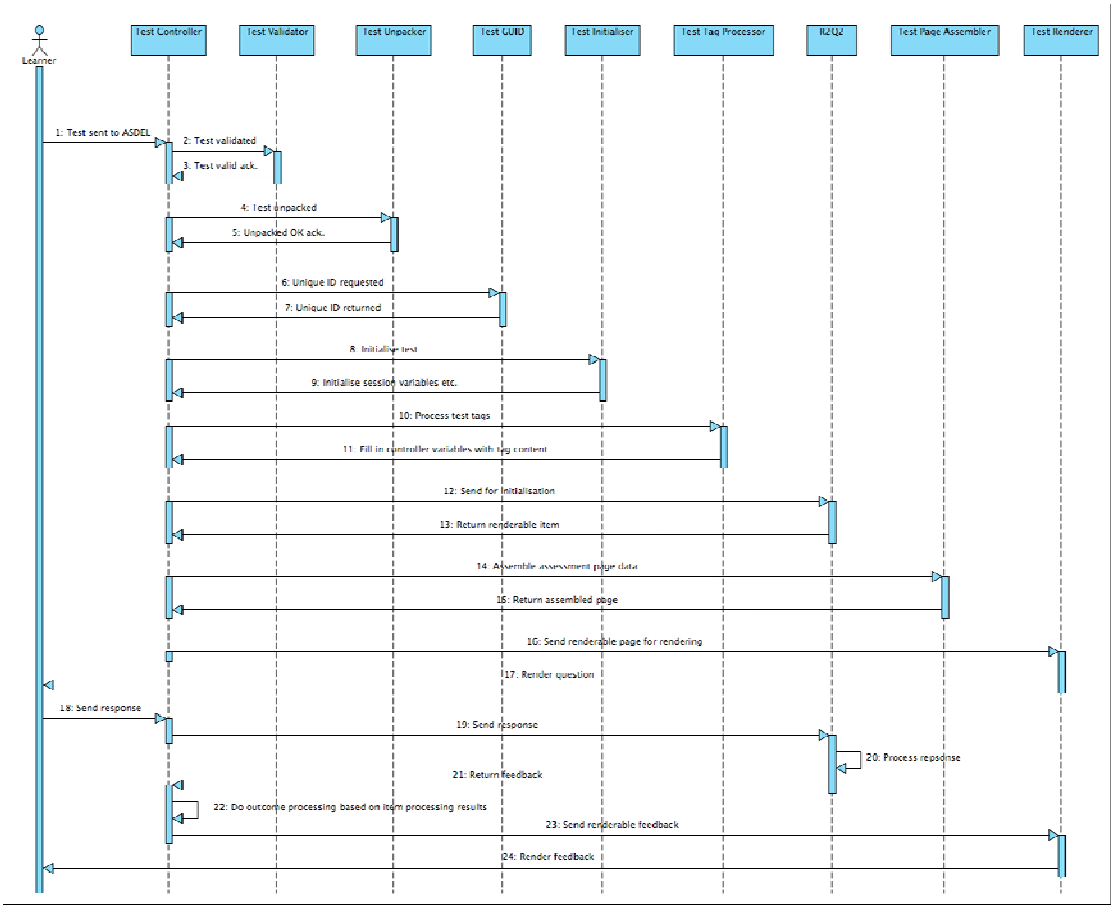

Figure 5: Typical sequence of events within the ASDEL system

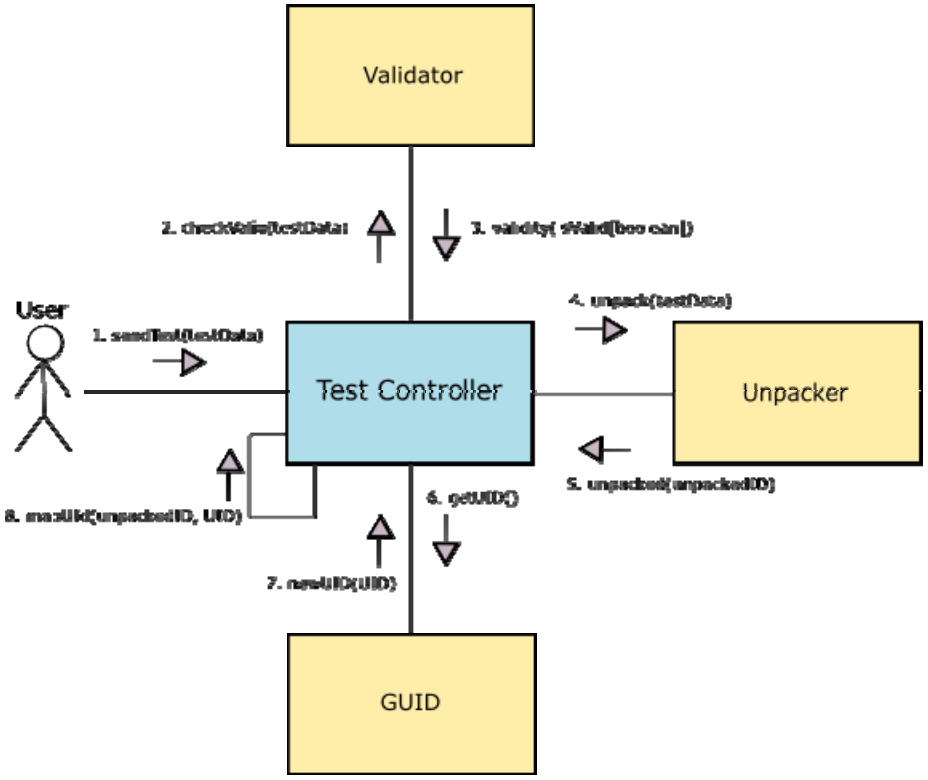

Figure 6: Collaborations between components during initialisation of a test

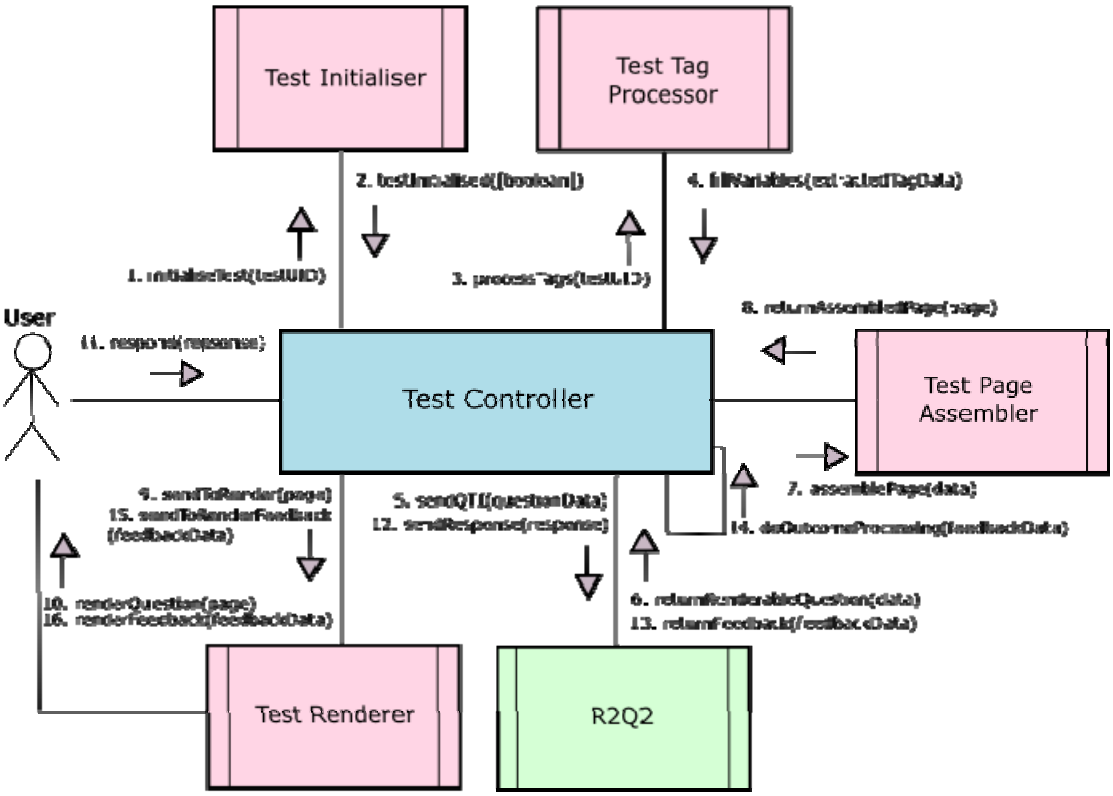

Figure 7: Collaborations between components as a test is undertaken

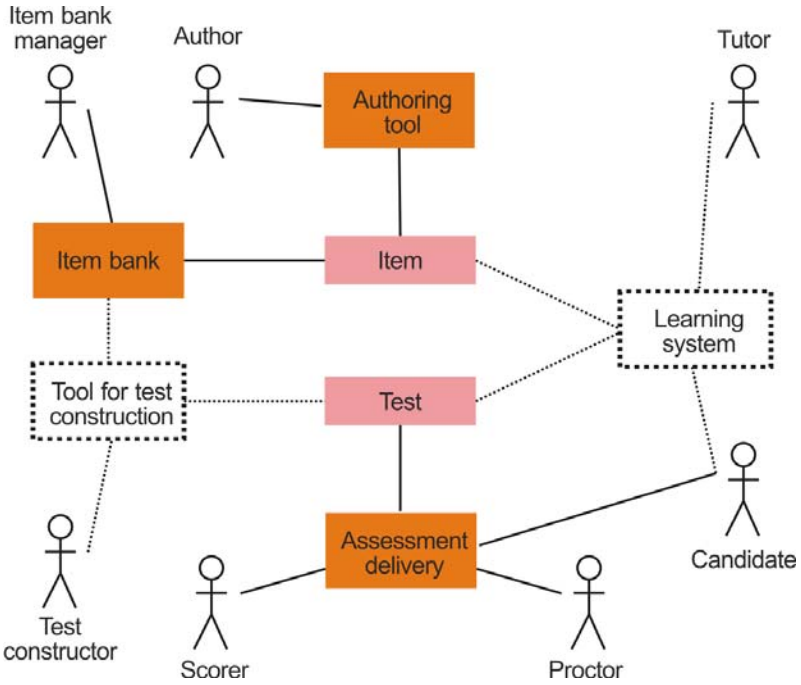

Figure 8. Phase Two: Integration of the ASDEL, AQuRate Item Authoring (Kingston) and MiniBix, Item Banking (Cambridge).

In the second phase, the project will integrate with the other projects in the JISC Capital Programme call on item banking (Cambridge: Minibix) and item authoring (Kingston: AQuRate) to provide a demonstrator, and will contribute to its evaluation and the evaluation of the project.

Figure 8 shows a modified diagram of the Use Case from the QTI v2 specification, demonstrating how the different tools and system in this call relate together. It clearly shows the boundaries between the delivery system, authoring tool, and item banking. A general scenario would be:

1. A lecturer/tutor will write questions (items). The authoring tool will provide a user interface appropriate to the end user, and format and store the items using the QTI v2 standard. By using QTIv2 these items may be exchanged with other compliant systems not developed by the same developer.
2. Users can select items from the item bank and place the items in a pool ready for constructing into a test. The test construction system, like the item authoring tool, will use an appropriate user interface and behind the scenes output the test in a QTI v2 or IMS CP compliant format.
3. By having the test and item adhere to the QTIv2 specifications, the deployment of items, item banks, and tests from diverse sources can be delivered through the test delivery system to candidates via a leaning environment or directly via their internet browser.
4. The candidate can now take the test, and have the results reported in a consistent manner.

The integration in this workpackage may be best achieved by using a using a portal framework to integrate the different projects.

5.1 Changes to R2Q2

During the design and implementation of ASDEL a number of issues have been identified in R2Q2 that will need to be fixed before the implementation is complete. Firstly, the default R2Q2 renderer renders full xhtml pages rather than rendering fragments. ASDEL requires fragments so that it can append various elements of rubric and other textual information about the test before and after the question. In the bigger picture, the output from ASDEL also needs to be in the form of a fragment so that it can be integrated with a VLE or portal framework. The second issue is that R2Q2 will always render the feedback that is included in an item. The problem is that the QTI assessment specification allows the delivery engine to control whether or not the feedback from an individual item should be delivered.

6. Conclusions

At a recent conference, the UK assessment community confirmed that kick-starting the use of the IMS Question and Test Interoperability version 2 specifications was a high priority. Whilst earlier versions of the specification provided most of the functions needed by practitioners, to ensure future interoperability it was considered essential that tools migrate to this new standard. However there was little incentive to move towards the new

specification as existing public implementations are incomplete. The conference concluded that there needed to be a robust set of tools and services that conformed to the QTIv2 specification to facilitate this migration.

R2Q2[5] is a definitive response and rendering engine for QTIv2 questions. While this only deals with an Item in QTI terms, it is essential to all processing of QTI questions; that is, it forms the core component of all future systems. Due to the design and use of internal Web services, the system could be enhanced if required. So while every effort has been made to ensure this service can be dropped into future systems, if necessary it can be changed to suit any application. The R2Q2 rendering and response engine of QTIv2 questions is expected to help two main stakeholders:

- ***Early adopters of QTIv2*** have written questions to this specification and need to validate the question. To help them we have provided a Web client to which they can submit questions and see the rendered version.
- ***Other e-Framework Projects***. We have provided the core elements of QTIv2 appropriate to a service oriented architecture. Applications in the area of e-assessment, and other aspects of the specification, need to be developed. The R2Q2 project would be an essential element in such future work.

In the ASDEL project we aim to build an assessment delivery engine to the IMS Question and Test Interoperability version 2.1 specifications. Like R2Q2 this will be a Web service based system that can be deployed as a stand-alone web application or as part of a Service Oriented Architecture enabled Virtual Learning Environment or portal framework. The engine will provide for:

- Delivery of an assessment consisting of an assembly of QTI items, with the possibility that the assessment is adaptive and the ordering of questions can depend on previous responses,
- Scheduling of assessments against users and groups,
- Rendering of tests and items using a web interface,
- Marking and feedback,
- A web service API for retrieving assessment results.

6.1.1 References

[1] Bull, J., and McKenna, C. Blueprint for Computer Assisted Assessment. Routledge Falmer, 2004.

[2] Conole, G. and Warburton, B. "A review of computer-assisted assessment". ALT-J Research in Learning Technology, vol. 13, pp. 17-31, 2005.

[5] http://www.r2q2.ecs.soton.ac.uk/.

[3] Cooper, A. and Reimann, R. About Face 2.0: The Essentials of Interaction Design. John Wiley & Sons, 2003.

[4] Davies, W. M., Howard, Y., Millard, D. E., Davis, H. C. and Sclater, N. Aggregating Assessment Tools in a Service Oriented Architecture. In Proceedings of 9th International CAA Conference, Loughborough. 2005.

[5] Olivier, B., Roberts, T., and Blinco, K. "The e-Framework for Education and Research: An Overview". DEST (Australia), JISC-CETIS (UK), www.e-framework.org, accessed July 2005.

[6] Sclater, N. and Howie K. User requirements of the ‘‘ultimate’’ online assessment engine, Computers & Education, 40, 285–306 2003.

[7] Wilson, S., Blinco, K., and Rehak, D. Service-Oriented Frameworks: Modelling the infrastructure for the next generation of e-Learning Systems. JISC, Bristol, UK 2004.

[8] APIS [Assessment Provision through Interoperable Segments] - University of Strathclyde–(eLearning Framework and Tools Strand) http://www.jisc.ac.uk/index.cfm?name=apis, accessed 30 April 2006.

[9] Assessment and Simple Sequencing Integration Services (ASSIS) – Final Report – 1.0. http://www.hull.ac.uk/esig/downloads/Final-Report-Assis.pdf, accessed 29 April 2006.

[10] IMS Global Learning Consortium, Inc. IMS Question and Test Interoperability Version 2.1 Public Draft Specification. http://www.imsglobal.org/question/index.html, accessed 9 January 2006.

[11] McAlpine, M. Principles of Assessment, Bluepaper Number 1, CAA Centre, University of Luton, February 2002.

[12] Draper, S. W. Feedback, A Technical Memo Department of Psychology, University Of Glasgow, 10 April 2005: http://www.psy.gla.ac.uk/~steve/feedback.html.