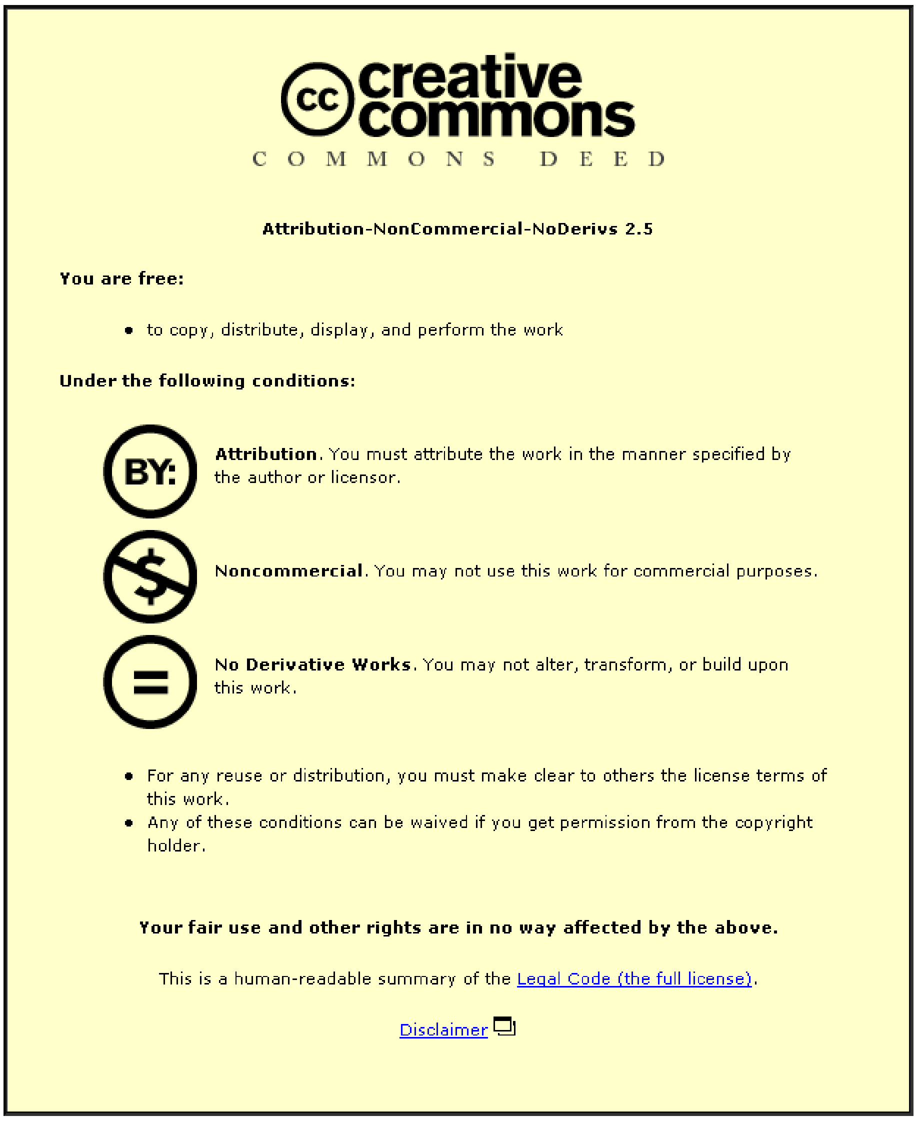

ISSUES RAISED DEVELOPING AQURATE (AN AUTHORING TOOL THAT USES THE QUESTION AND TEST INTEROPERABILITY VERSION 2 SPECIFICATION)

Alsop, G., Annesley J., Cai, Z., Campos, A., Colbert, M., Livingstone D., Smith, G. and Orwell, J.

Issues Raised Developing AQuRate (An Authoring Tool That Uses the Question and Test Interoperability Version 2 Specification)

Alsop G., Annesley J., Cai Z., Campos A., Colbert M., Livingstone D., Smith G., and Orwell J.
Learning Technology Research Group
Kingston University

Abstract

The IMS Question & Test Interoperability (QTI) specification has existed for many years, and there are a few tools for authoring questions in early versions of the specification. However, the new QTIv2 specification was unsupported in any existing authoring environment. The AQuRate project was funded by JISC's capital project program to fill this gap. AQuRate is one of three JISC projects, which together aimed to support the whole e-assessment process, from authoring (AQuRate at Kingston University) to storage (Minibix at Cambridge) and finally to a delivery/assessment development (ASDEL at Southampton). This paper considers issues raised during the creation of the tool: data modelling, graphical user interface design, and use cases. It ends raising issues currently effecting on-going development.

Data Modelling and the QTI Specification Version 2

The QTI Specification

The Question & Test Interoperability (QTI) specification describes a data model for the representation of question (assessmentItem) and test (assessment) data and their corresponding results' reports. Therefore, the specification enables the exchange of this item, assessment and results data between authoring tools, item banks, learning systems and assessment delivery systems. This specification has an open-ended, platform independent structure. It allows maximum flexibility and exchangeability.

The following diagram shows AQuRate's boundaries with its related projects.

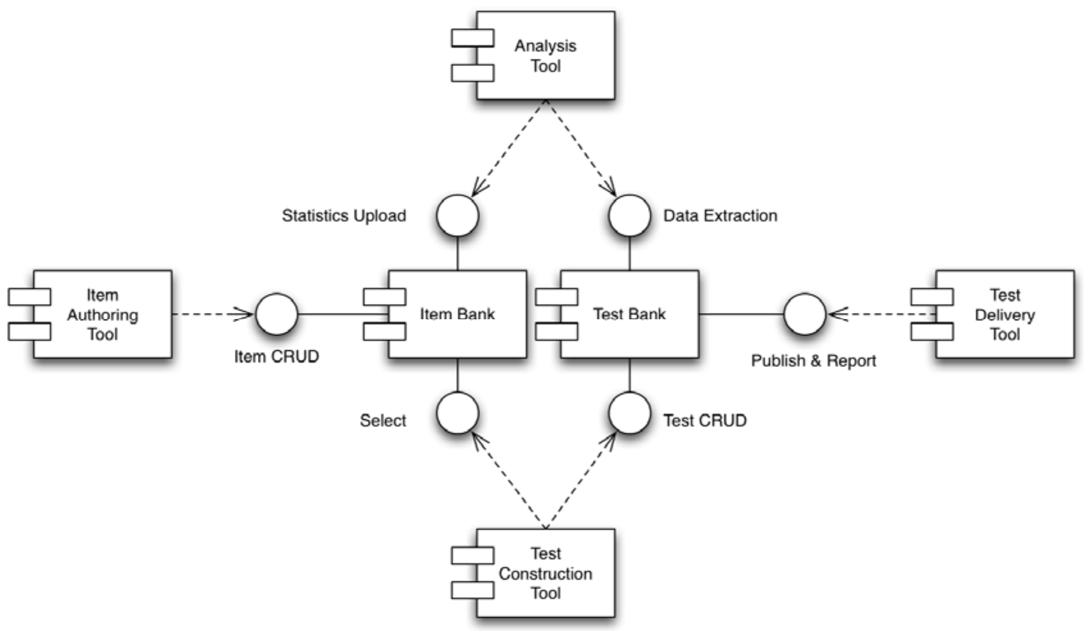

Diagram 1 courtesy of Steve Lay CARET

QTI Authoring

It was essential that our tool presents QTI features completely and coherently. However, given the allocated time and resources, it is very difficult to cover the entire scope with a single desktop application. Furthermore, application development was limited to question types that were identified as those most frequently used by authors (choice, associate, order, inline choice, slider, hotspot, graphic order and text entry).

This project used a Rapid Application Development (RAD) approach (Dynamic Systems Development Method DSDM). As a result users' requirements were constantly in focus. This led to some decisions on how to interpret certain features of QTIv2 in a particular way for implementation.

1. For example, "assessmentItem" is on the top level of the Extensible Markup Language (XML) structure for QTIv2. However, "AssessmentItem" does not contain any question type definitions. The end-user interaction with a question is defined by one or more "interaction" elements and their corresponding "response" elements inside the question. But the notion of a "type of question" is essential to convey the purpose of the application to new users. It was decided to define a question type by the interaction type contained in the question.
2. Furthermore, "responseProcessing" provides the ability to attach partial scoring, customized feedback etc. to an "interaction" or a particular answer. The complexity of "response processing" grows as the number of "interactions" (question types) and "choices" (answer options) grows. The balance has to be found between the overall usability of the tool and its ability to process arbitrarily complex items. This element of flexibility is both a strength and a weakness of the QTI specification for both developers and authors. The resulting range of valid questions is a strength of QTIv2. However, it is difficult for new users to understand how

to configure powerful response processing features to meet their needs. To be useful, then, it was necessary to offer users a defined range of available of available ‘options’ (a subset of valid QTIv2).

To deal with 1 and 2 for users the user interface of the tool blends 'wizard-like’ and ‘document-based’ design templates. Users are presented with a question document, and they may edit any attribute of this question. However, there is a clear sequence to the document, which guides novice users through that question type, and which experienced authors can use to work systematically.

The user interface encourages authors to explore the educational potential of QTIv2, and so learn to write more complex, and educationally more valuable questions. The tool does this in a number of ways: (i) a ‘sample question’ of each type is distributed with the tool; (ii) the ‘New Question’ dialog displays a description, example and application hints for each question type; (iii) a ‘Preview’ feature allows authors to see for themselves how learners will experience unfamiliar question types; (iv) document-based interface. Users can ‘play around’ with unfamiliar attributes (e.g. ‘shuffle’) one at a time and see the implications of changing values using the ‘Preview’ feature. There is no need to repeat all the steps in a Wizard just to see what ‘shuffle’ means.

For more technically able authors an XML version is available via a tab that allows direct editing of a question.

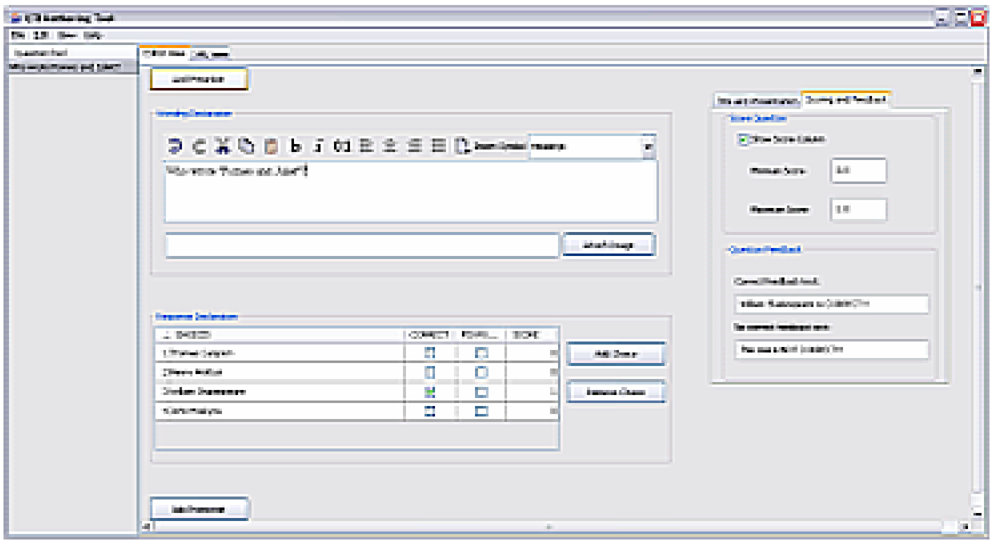

Diagram 2 AQuRate GUI

1. Using XML for describing QTI also raises issues regarding validation of any questions authored. Both syntactic and semantic validation is necessary, but the XML schema can enforce only syntactic validation. Semantic validation is mostly left to either the authoring tool or the delivery/assessment tool to enforce. For example, some response processing meta-data have to be in the question item, when certain conditions exist, although their presence cannot be mandated by the

schema (because there is other conditions in which they need not be present). Furthermore, if authoring tools and assessment/delivery tools are developed by different individuals, with different understandings of the semantic structure of QTI, some inconsistency between the authoring tool and the delivery/assessment tool might arise. So documentation needs to clarify the understanding. Should tools accommodate for alternative possible understandings?

Data Modelling

The base java technology used to implement QTI data model was Java Architecture for XML Binding (JAXB). It provided direct translation of the data model from the XML. It translated XML representations of QTIv2 questions to actual java classes. It also provided real-time syntactical validation of the QTI questions. JAXB allowed complete mapping from XML schema components to java classes. Some may argue that the java structures generated by JAXB are too fine-grained, with the consequence that the large number of automatically generated java classes is unmanageable for software developers, who would prefer to work with a smaller number of more complex classes. However, it gives sufficient flexibility to implement the authoring tool in a RAD project. It also allows rapid reimplementation of changes as the QTIv2 specification evolves.

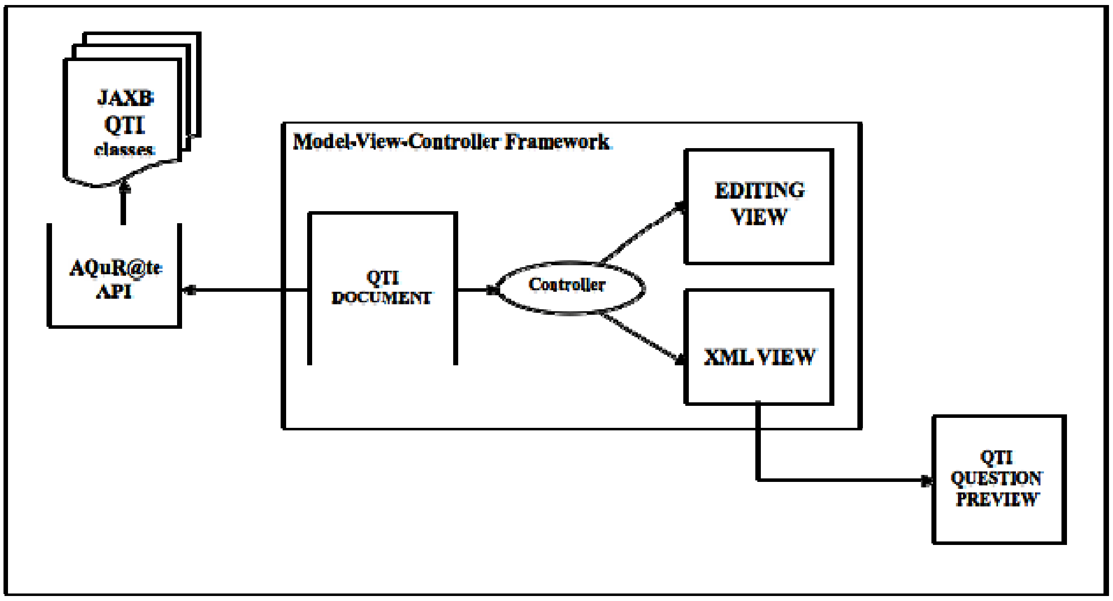

Diagram 3 Architecture

Use Case Issues

As is often the case with projects of this nature, the software development process exposes details with (and alternatives to) the original ‘use-case’ scenario. Three such issues are discussed below: the rendering of content in an authoring environment; the status of collections of items in this authoring environment, and the opportunity to provide material for an alternative ‘standalone’ delivery scenario.

The rendering of content

One requirement for the Desktop Authoring Tool is for authors to be able to render the questions in ‘the same’ presentation format that would be used to present the material to candidates. To provide this function, a software library is used to convert the XML into HTML, load this into a browser, and then process the user response, and provide the appropriate feedback. However, there are several reasons why this cannot be considered the ‘definitive’ presentation of the content. Firstly there are the well-known differences between the browsers available on different operating systems. Secondly, there are the (less-well known) differences in output, between the available rendering engines. Depending on the rendering engine used at the point of delivery, the presentation may have some differences, compared to what is provided to the user. This will be more likely in ambitious or non-standard editing approaches, and hence the standardization of the formatting structures used by the editor may be necessary to achieve the required uniformity of output. Finally, it must be recognised that some delivery systems may impose their own presentational structure, such as an institutional style-sheet, or a style sheet designed to make the content accessible for an individual with a visual impairment. It is now clear that these are details which must be addressed in future versions of the authoring tool.

The status of collections of items

The authoring tool was originally specified to allow several QTI items to be loaded or generated concurrently. The author is able to switch between the items, e.g. for reference, or for cutting and pasting specific material between them. However, it is now clear that the capability of manipulating several QTI items concurrently has implications for test authoring as well as item-authoring. At its most simple, the collection of items could be exported as a simple linear sequence of questions. More generally, the relations between the elements of the collection could be defined more precisely, to enable adaptive testing. However, it is not clear how this would stand in relation to the normal community usage. In formal assessment contexts, authoring of items is a different activity to authoring of tests. In less formal contexts (e.g. formative assessments in higher education, or accompanying material for textbooks) the two activities are often combined, and so this functionality would be useful.

Standalone delivery

Originally, the authoring tool was part of a large scale use-case scenario including an item repository and a web-based delivery system. Furthermore, this scenario was not complete: it also required integration with a Virtual Learning Environment (VLE) and the appropriate authentication procedure. However, the creation of a standalone (‘Desktop’) rendering system provided the opportunity to generate a self-contained set of items and software to process item responses. This could be made available to students as a folder of HTML files (and auxiliary content) that they could use with a browser, but without an internet connection. This alternative use-case scenario may be useful in situations where internet connection is difficult to arrange (or guarantee). It also has the benefit of providing pre-rendered material, thus reducing the uncertainty about how the content will be presented to the user.

Conclusions

The decisions made in interpreting the specification in order to limit the complexity of development were useful in ensuring that a tool was developed in the time available, but this did lead to a trade off in design.

Due to the different possible semantic interpretations of the specification in developing authoring and delivery tools, documentation is essential to limit misunderstandings.

The use of JAXB sped the development of the tool and allows for easy reimplementation as the specification for QTI evolves.

The complexity and coherence of QTIv2 aids software development to support it, but ironically might phase some question authors. Thus there was a need to create a tool that worked for both novices (using wizards) and an xml edit view for experts. This complexity also applies to the educational design of questions. Here, offering a limited set of questions did not necessarily advance the creation and development of new opportunities available through the new specification.

The use of a desktop tool with a built in renderer allowed for the fast testing of questions and use of the tool for formative assessment purposes with and without the need for an item bank or assessment delivery engine. This allows for widespread use.

The relationship between question and test authoring needs careful consideration. The JISC programme funding the three related projects covered an item authoring tool, item bank and assessment delivery engine, but not a test/assessment authoring tool. These activities are often combined and the rationale for separation needs more thought.

Finally, the standalone use case led to a useful and unexpected outcome of a Desktop rendering tool that may lead to a use in formative assessment.

Questions for debate

Should simplicity or complexity guide the data model if an outcome of simplicity leads to limitations of educational use?

Should an authoring tool using a new standard cater for novice or expert users? Focussing on the expert may lead to a tool that shows the potential of the standard rather than a tool for standard users.

GUI design accommodated both novices and experts, but was governed by a user selecting a type of ‘question item’ to author. This led to an avoidance, for example, of nesting questions within ‘multiple interactions’. Should users be ignored when new things are possible?

Will standalone delivery be a useful option that is worthy of future development?

Acknowledgements

The Joint Information Systems Committee (JISC) funded the work.

References

AQuRate QTI Ver2 Authoring Tool http://aqurate.kingston.ac.uk, accessed 7 March 2008

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.

IMS QTIv2 Examples, Powered by JAssess, http://qtitools.caret.cam.ac.uk/qtiv2/examples/V2imsexamples.html, accessed 29 February 2008

Sun Java Architecture for XML Binding (JAXB) http://java.sun.com/developer/technicalArticles/WebServices/jaxb/, accessed 5 March 2008

QTI Training Guide by JISC http://wiki.cetis.ac.uk/QTI_Training_Guide, accessed 29 February 2008

Questionmark Perception QTI Authoring http://www.questionmark.com/us/perception/authoring.aspx, accessed 4 March 2008