<e-QTI>: a Reusable Assessment Engine

Iván Martínez-Ortiz[#], Pablo Moreno-Ger[*], José Luis Sierra [*], Baltasar Fernández-Manjón[*]

# Centro de Estudios Superiores Felipe II. 28300, Aranjuez. Spain
imartinez@cesfelipesegundo.com

* Dpto. Sistemas Informáticos y Programación. Fac. Informática. Universidad Complutense. 28040, Madrid. Spain.
{pablom,jlsierra,balta}@fdi.ucm.es

Abstract. <e-QTI> is a highly modular and extensible engine that simplifies the assessment cycle in terms of generation, execution, presentation, grading and archiving. <e-QTI> is based on the IMS QTI specification but it is able to export and import assessments represented in a wide range of formats, and also to maintain a pool of questions that can be reused in a wide variety of authoring situations. The engine is especially suited to be integrated in web-based learning platforms using well-defined interfaces, both Java-oriented and service-oriented. In this paper we describe <e-QTI>, its internal architecture and its most relevant implementation details.

1. Introduction

Traditionally, teachers have made use of exams to evaluate learners during the educational process as a requirement for the fulfilment of a course or simply as a measurement of the assimilation of concepts presented in a lecture/class/course. This assessment style has also been adopted in the e-learning context of modern LMSs (*Learning Management Systems*) [3,13], which usually include in their toolsets an assessment tool that not only enables the teacher to create the exams but also facilitates the grading task since many types of questions can be evaluated automatically. For this purpose, the LMSs must face several technological challenges. On the one hand, the significant variability in the formulation of the exams must be addressed. Typical exams can include many different types of questions (e.g. essays, true/false, multiple choice, matching/ordering lists, hot-spots, fill in the gaps, etc), and these questions can involve many different content formats (e.g. text, images, audiovisual content, etc). On the other hand, many different evaluation strategies (e.g. automatic correction, different grading algorithms, instructor intervention, etc) must also be contemplated. Moreover, from the learner's perspective, exams must accommodate many different solution styles. Finally, exams must be represented in standard formats that ensure their independence from any given LMS, thus promoting their durability and portability, and also the reusability of individual questions in the formulation of new exams. The correct management of individual questions can facilitate the production of exams on-the-fly by an intelligent LMS in order to evaluate the skills of a particular learner in the context of custom learning paths.

In this paper we describe <e-QTI>, an assessment engine that addresses most of the aforementioned challenges. The architecture of <e-QTI> is driven by the IMS QTI (*Question and Test Interoperability*) specification [19], and the engine provides native QTI support. Nevertheless, this engine is very flexible and can be extended with pluggable import/export facilities, therefore being able to accept and produce exams in many other different formats. <e-QTI> has been developed in the context of our <e-Aula> experimental LMS [1,15,34]. However, it is an independent, robust and open tool that can be used standalone or integrated in different learning architectures or applications.

The structure of the paper is as follows: Section 2 describes some of the aspects of IMS QTI needed to understand the rest of the paper. Section 3 describes the basic functionalities of the <e-QTI> engine. Section 4 details the internal architecture of this engine. Section 5 presents some related work. Finally, section 6 gives some conclusions and outlines lines of future work.

2. IMS QTI

IMS QTI is an IMS specification for enabling the exchange of data related to questions, tests and results of the assessment process between heterogeneous IMS-compliant systems and tools [19]. This interchange is usually performed using a binding of the IMS QTI abstract models with XML (*eXtensible Markup Language*) [9]. In this section we summarize the concepts of QTI that are relevant for the present work.

In QTI jargon, questions are called *items*, while tests/exams are called *assessments*. QTI also introduces an intermediate structuring construction called *section*, which is used to group individual items or other simpler sections together. In addition, IMS QTI defines the concept of *object bank*. Objects banks are used to package a set of individual items or sections to be exchanged between LMSs or question repositories to create question pools, so that teachers can reuse these pools when creating new exams.

QTI also enables the declarative description of many relevant dynamic features of the assessment process. For each structural element it is possible to attach a set of *rules* used to process learner responses. These rules can, for instance, trigger the presentation of some feedback. Their formulation can be based on a set of control switches, which are also included in the description of the corresponding structural element or their neighbors. In addition, with sections and assessments it is also possible to attach a set of *selection* and *ordering* rules enabling instructors to decide the sequence in which sections and/or items are presented to the learner and also how he/she can interact with them. Finally, IMS QTI also enables the description of how the aggregate scores at the assessment and section levels can be derived. This derivation is described using *scoring rules,* which establish how to synthesize the score of an assessment/section from the scores of their child sections and/or items. It is important to note that scoring rules do not specify the internal details of particular scoring algorithms. Instead, the type of algorithm is specified and the necessary parameters are supplied. The implementation and execution of the corresponding algorithm is relayed to the assessment engine. Although QTI defines the most common types of scoring algorithms and their semantics, the specification does not force the use of any of these, and therefore other domain-specific algorithms can also be readily supported by a particular engine.

Finally, IMS QTI is related to other IMS specifications. In particular the IMS Content Packaging (IMS CP) [17] and IMS Metadata / IEEE LOM [16] specifications are closely related to <e-QTI> development. IMS CP, which defines how to aggregate the educational contents into packages in order to let different heterogeneous systems interchange these contents, is used in <e-QTI> as a mechanism for packaging items, sections, assessments and object banks with their related files. In its turn, IEEE LOM metadata is used in <e-QTI> to classify items and sections from an educational point of view.

3. The <e-QTI> engine

<e-QTI> is a highly modular and extensible QTI assessment engine. The engine allows instructors to author QTI assessments, as well as deliver them and make them accessible to learners. Instructors can also import and export assessments from/to other representation formats, and store and retrieve items in/from a question pool. In this section we describe the main functionalities of the engine, while the architectural and implementation details are relegated to the next section.

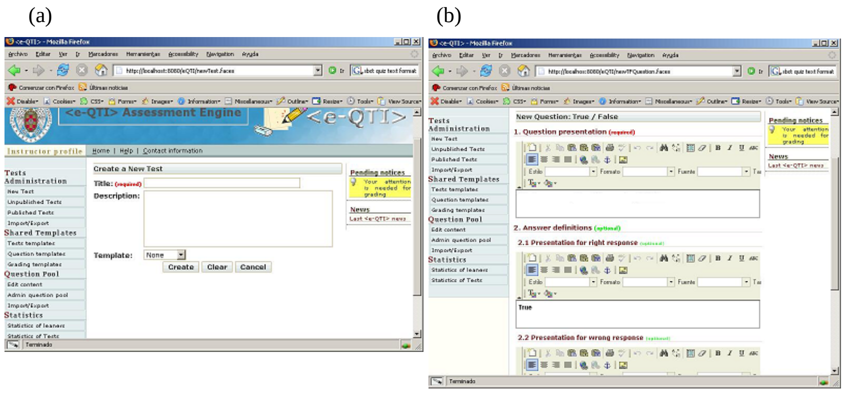


Fig 1. (a) <e-QTI> authoring of a new QTI Test; (b) Edition of an True/False item of a Test

3.1. Authoring

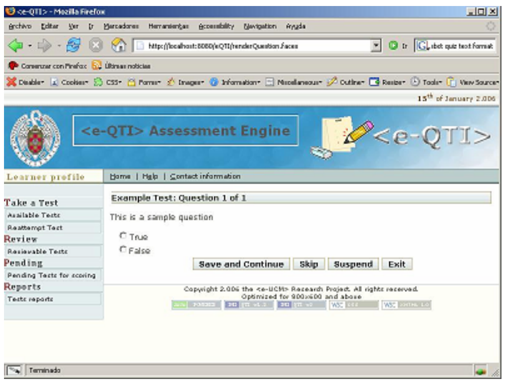
<e-QTI> allows instructors to produce and maintain QTI assessments by using a special-purpose editor, therefore relieving them of the complexities behind QTI XML

binding. Instead of coping with XML files (perhaps using a general-purpose XML editing tool, like [23]), instructors can use the more user-friendly and domain-specific authoring artifacts provided by <e-QTI> (Fig 1a). This way, acceptability and productivity are greatly increased, as we have realized during our preliminary experiences with <e-QTI> in the Complutense University of Madrid.

The authoring process in <e-QTI> is driven by the QTI information model. Using <e-QTI>, the author can create, edit and delete assessments and their different inner elements. The edition of each informational element comprises the edition of all its relevant properties. Each item may be associated with suitable presentation materials, definitions of objectives and processing rules, and selection and ordering rules. Finally the authoring system also permits the recovery of items from the question pool maintained by the engine (see subsection 3.4).

The authoring of items is the main part of the authoring system (Fig 1b). An item comprises several aspects that can be edited. This amount of information can overwhelm the instructor, so the <e-QTI> authoring tool deals with this problem by offering a more comfortable, more friendly and well known user interface based upon of the type of question. <e-QTI> now supports the authoring of the following question types: *True*/*False*, *multiple choice*, *fill in the gaps*, *matching lists*, and *essays*.

Finally <e-QTI> offers the possibility of adding more detailed information to items, sections and assessment by allowing the instructor to add metadata information, objectives and rubrics.



<e-QTI> - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Accessibility Navigation Ayuda
http://localhost:8080/eQTI/renderQuestion.faces
Comenzar con Firefox Últimas noticias
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source
<e-QTI> Assessment Engine
<e-QTI>
Learner profile
Home | Help | Contact information
Take a Test
Available Tests
Reattempt Test
Review
Reviewable Tests
Pending
Pending Tests for scoring
Reports
Tests reports
True
False
Save and Continue
Skip
Suspend
Exit
Terminado

15th of January 2.006

Example Test: Question 1 of 1

This is a sample question

Copyright 2.006 the <e-UCM> Research Project. All rights reserved.
Optimized for 800x600 and above

Fig 2. Presentation of an item made by <e-QTI>

3.2. Delivery

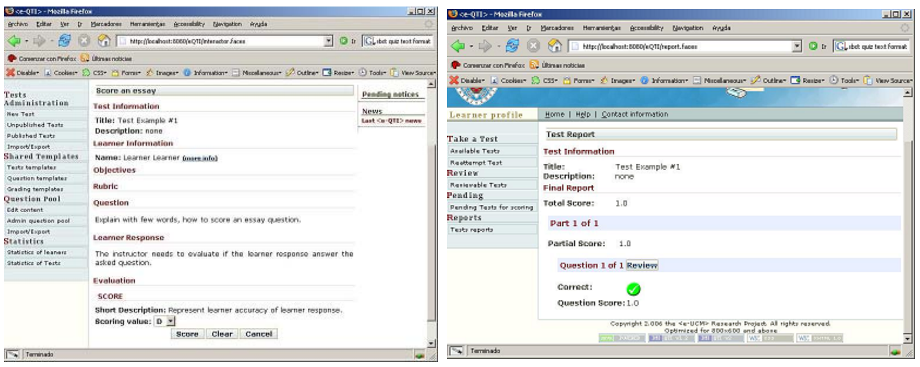
<e-QTI> also controls the presentation of the assessments to the learner, as well as the interaction between the learner and these assessments (Fig 2). The engine supports the interaction with learners while they are taking the test, displaying relevant information such as time remaining in a time-fixed test or allowing him/her to partially complete the assessment and finalize it later. It also allows learners to review the evaluation and displays feedback materials added by the instructor once the test or survey has been graded.

The behavior of the <e-QTI> engine during delivery is heavily driven by the processing rules attached to the assessment's structural elements (the assessment itself, its sections and its items), and also by the selection and ordering rules attached to this assessment and their sections. Therefore, the engine lets instructors customize this behavior to better fit the special needs of each particular learning scenario.

3.3. Grading

<e-QTI> engine gives support to the evaluation of assessments completed by learners. The engine is highly flexible and extensible, in order to accommodate different evaluation styles. These styles range from the automatic correction of simple items to the explicit involvement of the instructors when human assistance is required (Fig 3a). For example, <e-QTI> delegates evaluation of essays to the corresponding tutor, so global evaluation needs to wait for the response of the tutor. Finally, the engine also presents the final report to the learners (Fig 3b).

(a) (b)



Score an essay

Test Information

Title: Test Example #1
Description: none

Learner Information

Name: Learner Learner (more info)

Objectives

Rubric

Question

Explain with few words, how to score an essay question.

Learner Response

The instructor needs to evaluate if the learner response answer the asked question.

Evaluation

SCORE

Short Description: Represent learner accuracy of learner response.
Scoring value: 0

Test Report

Test Information

Title: Test Example #1
Description: none

Final Report

Total Score: 1.0

Part 1 of 1

Partial Score: 1.0

Question 1 of 1

Correct: ✓
Question Score: 1.0

Copyright 2006 the <e-UCM> Research Project. All rights reserved.
Optimized for 800x600 and above

Fig 3. (a) Interaction with instructor during grading, (b) final report presented to the learner

As with delivery, grading in <e-QTI> is highly configurable by instructors. The engine makes use of the grading rules included in sections and assessments in order to produce the final scores. As described in section 4, the grading algorithm can be chosen from the available algorithms and bound to an assessment or section. The predefined algorithm templates offered by IMS QTI are the ones most commonly used by the e-learning community, but the test creator usually needs to provide a set of values to instantiate the template. <e-QTI> offers the possibility of instantiating preconfigured templates so that the creator does not need to know the meaning of all parameters, which is particularly useful when applying organizational grading conventions. Also, it is possible for the instructor to tweak the grading algorithm parameters as needed.

3.4. The Question Pool

<e-QTI> maintains a question pool containing items that can be used while authoring new assessments. The system allows the navigation of this pool in order to retrieve items during the authoring process. <e-QTI> also offers the possibility of adding new items to the question pool. To facilitate later search and navigation, the question pool is organized by using a classification system. Currently, <e-QTI> uses a simple classification, fixed by the system administrator. Nevertheless, he/she can easily edit and change this hierarchy when necessary. This classification information is added to the item or section using the IEEE LOM classification category.

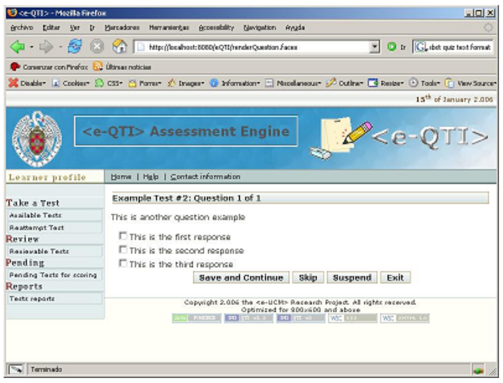
3.5. Importation/Exportation

<e-QTI> allows instructors to import and export exams from/to the IMS QTI format. This feature is especially valuable in turning <e-QTI> into a tool capable of working with third party applications, like authoring tools such as Canvas Learning [10] or Respondus [31]. This feature allows authors to use other alternative authoring tools in the creation of assessments that can be played in <e-QTI>.

(a)

```
:TYPE:MC:N:C
:TITLE:This is another
question example
:QUESTION:T
This is the question content
itself
:ANSWER1:100:T
This is the first response
:ANSWER2:0:T
This is the second response
:ANSWER3:0:T
This is the third response
```

(b)


<e-QTI> - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Accessibility Navigation Ayuda
http://localhost:8080/eQTI/renderQuestion.faces
Comenzar con Firefox
Últimas noticias
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source
<e-QTI> Assessment Engine
<e-QTI>
Learner profile
Home | Help | Contact information
Take a Test
Available Tests
Reattempt Test
Review
Reviewable Tests
Pending
Pending Tests for scoring
Reports
Tests reports
Save and Continue
Skip
Suspend
Exit
Terminado

15th of January 2.006

Example Test #2: Question 1 of 1

This is another question example

- ☐ This is the first response
- ☐ This is the second response
- ☐ This is the third response

Copyright 2.006 the <e-UCM> Research Project. All rights reserved.
Optimized for 800x600 and above

Fig 4. (a) WebCT multiple choice quiz format (b) <e-QTI> preview

The system itself is extensible by nature in order to accommodate other formats that can be reinterpreted and mapped using the extensibility features provided by <e-QTI>. For example, <e-QTI> now supports the WebCT [30,38] quiz format (Fig 4). We have also included facilities to import and export assessments bundled according to the IMS CP specification.

4. Architecture of <e-QTI>

The current <e-QTI> tool is an evolution of a previous prototype embedded in our <e-Aula> research LMS. As we have realized during the maintenance of this former tool, since the QTI specification is open to changes, the system should be as flexible and scalable as possible.

The design of <e-QTI> was created while keeping in mind that it should be possible to integrate this tool as a whole into an LMS, so the LMS calls <e-QTI> when the assessment service is needed. This means that an internally complex monolithic engine should be avoided, instead promoting modularity in order to assure maintainability and facilitate the integration task.

The next subsections details the architecture of <e-QTI>. Subsection 4.1 gives an overview of this architecture. Subsection 4.2 summarizes how this architecture is implemented in <e-QTI> using well-known development standards.

4.1. Overview of the architecture

Fig 5 depicts the architecture implemented in the <e-QTI> tool, which we have been testing as a standalone application in several courses at the Complutense University of Madrid (Spain) as a support learning tool for the weekly evaluation of learners.

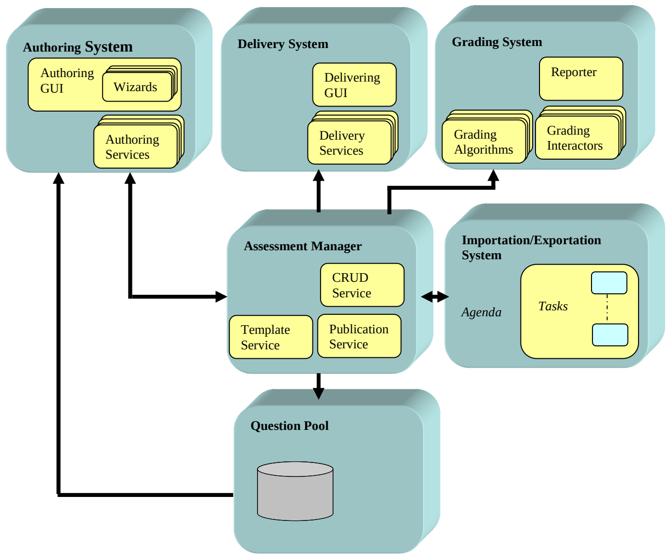


Fig 5. <e-QTI> Architecture overview

The <e-QTI> architecture covers all the requirements described in section 3 with a distinct system dedicated to each one of those functionalities. All of these systems are internally divided in a set of services to simplify the development of the entire engine. This division into services permits the modification, tweaking or entire re-writing of each system without affecting the rest of the application. The systems integrated in the architecture are:

– The *assessment manager*, which comprises all services needed to handle the <e-QTI> model of an assessment. The services included in the module are: (i) a CRUD (Creating, Retrieving, Updating, and Deleting) *service*, which supports the <e-QTI> model, (ii) a *publication service*, whose job is to process the authored assessment, take a snapshot of the current state of the assessment created and add any additional information needed before publishing, and (iii) a *template service*, which is in charge of the management and instantiation of the different types of templates used by <e-QTI>.
– The *delivery system*, which is intended to collect from the learners who interact with the tests and surveys proposed by teachers all the information required by other parts of the engine. This system is divided into two main components: (i) the *delivery* GUI, which renders the elements of the assessment by using the XSLT transformation language [12] to process the XML representation of the IMS QTI test, and (ii) the *delivery services*, which comprise all the functionalities needed to play the assessments, including the rearrangement of items and sections (using selection and ordering rules) and the extraction of information from the delivery GUI to be used later by the grading service. It is important to remark that the delivery services are independent of the delivery GUI. Therefore it is possible to change the GUI while maintaining the same set of services. Thus, a system that integrates <e-QTI> can use its delivery services but it can provide its own user interface.
– The *authoring system*, which is intended to create and edit assessments, sections and items used inside <e-QTI>, including elements that are part of the question pool. This system comprises the following elements: (i) an *authoring* GUI, which also includes a set of *wizards* that can be used in a fast and straightforward manner by the instructors, and (ii) a set of *authoring services*, which represent all the available commands that are necessary to author an IMS QTI test. As with the delivery system, authoring services are GUI independent. Therefore, third-party systems integrating <e-QTI> can provide their own authoring GUIs as long as these user interfaces are properly connected to the existing authoring services.
– The *grading system*, which processes the learners' responses gathered by the *delivery system*. It makes use of three mayor components: (i) a *reporter*, which is responsible for the final generation of the learner result report including all the automatic and manual evaluations of questions, (ii) a set of *interactors*, which are responsible for the interaction with instructors during grading operations, and (iii) the *grading algorithms* that can be used by the corresponding QTI's scoring rules.
– The *importation / exportation* system, which deals with the importation/exportation of assessments from/to other representation formats. To cope with the complexity and the potential variability of these tasks, we have adopted for this system an *agenda-based* organization similar to that proposed in [35] to architect the <e-Aula> importation system, which in turn is inspired by the solution described in [2] for the simulation of discrete systems.
– The *question pool*, which is offered as a central repository of questions to be shared between instructors that use <e-QTI>. This repository is organized internally as sections and items classified according to the classification hierarchy

mentioned in subsection 3.4. Thus, each classification entry corresponds to a section, so the classification is mapped onto a section hierarchy. This particular organization facilitates the exportation and importation of the content question pool using the IMS QTI format.

4.2. Implementation Details

<e-QTI> was developed using Java technologies in order to maximize the maintainability, extensibility and robustness of the resulting implementation. In particular, a multi-tier organization has been adopted, where the following tiers can be distinguished:

– *Client tier***.** We adopt a thin-client approach where we only need a web browser with XHTML, CSS and JavaScript support. This also facilitates the replacement of this tier in order to accommodate the GUI skin to particular authoring and learning scenarios.
– *Web tier***.** <e-QTI> uses a new Java technology called Java Server Faces [7] which facilitates the organization of this tier by using the classical Model-View-Controller (MVC) design pattern [25], as well as the development of reusable web components.
– *Business tier***.** It was created by using the POJO (*Plain Old Java Object*) approach [32] instead of adopting EJBs (*Enterprise Java Beans*) [8]. This allows <e-QTI> to be integrated in a web-based application by using a Java Servlet container, like Apache Tomcat, and also its business tier to be integrated inside a Java Desktop application. To facilitate the management of services and dependencies, <e-QTI> uses the Spring Framework [22], which greatly simplifies the creation of Enterprise Applications by using the POJO approach.
– *Persistence tier***.** <e-QTI> uses the Hibernate framework [6] inside the persistence tier to simplify the development of persistence onto a relational database. This choice decouples this tier from the particular database infrastructure. For instance, it is possible to use an in-process Java database to embed this tier inside a Java Desktop application in a transparent way.

<e-QTI> also exhibits a service-oriented programmatic interface [11]. The different services introduced in the conceptual architecture are exhibited as web services, and they can be effortlessly integrated in third-party e-learning systems using SOAP (*Simple Object Access Protocol*) regardless of their underlying implementation technologies.

5. Related Work

As has already been mentioned, it is common to find an assessment mechanism in the toolset of a LMS (e.g. WebCT/BlackBoard [30,38], Moodle [26] and others). The assessment system in these tools is completely embedded. Therefore it can only be used inside these tools. However, it is sometimes possible to import and export resources in some tools, although they often use a proprietary format. Nonetheless, there are renowned tools that actually support standardized formats, such as Moodle's multi-format support, which includes IMS QTI.

Regarding the architecture of the system, some LMS implementations are truly modular. In particular, SAKAI [33] includes the Samigo module as an assessment tool that can be extracted and used as a standalone application. This module supports IMS QTI specification in both versions (1.2 and 2.0). There are other projects such as Technologies for Online Interoperable Assessment (TOIA) [36] established in 2003 to provide an advanced online assessment management system free of charge to all UK Further and Higher Education institutions.

Web services have been successfully used to address the integration of QTI and other IMS-compliant functionalities. Among such initiatives, it is interesting to point out the ASSIS Project [5] of Hull University, which has succeeded in integrating an IMS QTI v2 player, a reduced version of the viewer developed by the APIS Project [4], with an IMS Simple Sequencing [20] engine. In the resulting integration, the engine is used to sequence the items presented to learners. A similar experience was adopted in the last version of the IMS Learning Design (IMS LD) [18, 24] engine Coopercore [14, 37], which integrates the APIS Viewer using services, so now an IMS LD Unit of Learning can launch an assessment service.

6. Conclusions and future work

In this paper we have described our effort at building <e-QTI>, an assessment engine, and the architecture that supports it and allows it to be easily integrated in different environments.

From the point of view of the users, <e-QTI> improves the usability of our previous prototypes due to its focus on simplicity demanded by most users by hiding the IMS QTI specifications details and making intensive use of templates to improve the productivity.

From the architectural point of view, <e-QTI> is based on a highly extensible and modular architecture. This architecture makes the extension and/or the replacement of the different GUI parts possible without affecting the rest of the components. Also new delivery and authoring services can be added and existing ones can be replaced with equivalent counterparts in a transparent way. This is also true for grading algorithms in the grading system, and also for importation and exportation tasks in the agenda-based importation/exportation system.

From the implementation perspective, the Java-based implementation of <e-QTI> allows us to create a robust architecture which, in spite of being implemented with plain Java objects, becomes powerful and scalable due to the use of powerful frameworks in their different tiers, like Spring and Hibernate. In addition, <e-QTI> adheres to a service-oriented approach, and therefore the engine is powered with a web service-based programmatic interface, which facilitates its integration with many other heterogeneous e-learning systems.

Currently <e-QTI> has been successfully integrated into the <e-Aula> LMS. As future work we are planning to integrate <e-QTI> into other environments. In particular, we are interested in the integration of the assessment engine into SAKAI and in exploring potential interactions with Coopercore. Our research line on the evaluation and implementation of potential standards will also explore other specifications such as the IMS Tools Interoperability Guidelines [21], Open Knowledge Initiative Open Interface Service Definitions (OKI OSID) related to the assessment management [27, 28] and also specifications more related to question-bank querying like Remote Question Protocol (RQP) [29].

Acknowledgements

The Spanish Committee of Science and Technology (projects TIC2002-04067-C03-02, TIN2004-08367-C02-02 and TIN2005-08788-C04-01) has partially supported this work.

References

1. <e-Aula> project. Prototype available online at http://eaula.sip.ucm.es (last visited on January 2006)
2. Abelson, H.; Sussman. G.J. Structure and Interpretation of Computer Programs - Second Edition. MIT Press. 1996
3. Avgeriou, P.; Papasalouros, A.; Retalis, S.; Skordalakis, M. Towards a Pattern Language for Learning Management Systems. Educational Technology & Society, 6(2), pp. 11-24. 2003
4. APIS. More information available at http://apis.sourceforge.net/ (last visited January 2006)
5. ASSIS. More information available at http://hull.ac.uk/esig/assis (last visited January 2006)
6. Bauer, C., King, G. Hibenate in Action. Manning Publications & Co. 2004
7. Bergsten , H. JavaServer Faces. O'Reilly. 2004
8. Bodoff, S.; Armstrong, E.; Ball, J.; Carson, D. The J2EE Tutorial, Second Edition. Addison-Wesley Professional. 2004
9. Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E (Eds), *Extensible Markup Language (XML) 1.0 (Third Edition)*. W3C Recommendation. 2004
10. Canvas Learning. More information available online at http://www.canvaslearning.com/ (last visited January 2006)
11. Cerami, E. Web Services Essentials. O’Reilly. 2002
12. Clark, J (Ed). XSL Transformations (XSLT) Version 1.0. W3C Recommendation. 1999
13. Collier, G. e-Learning Application Infraestructure. Sun Microsystems White Paper. 2002
14. Coppercore. More information available at http://www.coppercore.org (last visited January 2006)
15. Fernández-Manjón,B.; Sancho, P. Creating cost-effective adaptive educational hypermedia based on markup technologies and e-learning standards. Interactive Educational Multimedia 4. 2002
16. Hodgings, W. (chair). IEEE Standard for Learning Object Metadata. IEEE Standard 1484.12.1-2002. 2002

17. IMS Content Packaging Specification v1.1.4. 2004. Available online at http://www.imsglobal.org/content/packaging (last visited on January 2006)
18. IMS Learning Design 1.0. 2003. Available online at http://www.imsglobal.org/learningdesign (last visited on January 2006)
19. IMS Question & Test Interoperability v 2.0. 2005. Available online at http://www.imsglobal.org/question (last visited on January 2006)
20. IMS Simple Sequencing Specification 1.0. 2003. Available online at http://www.imsglobal.org/simplesequencing (last visited on January 2006)
21. IMS Tools Interoperability Guidelines v 1.0 Public Draft. Available online at: http://www.imsglobal.org/ti/ (last visited on January 2006)
22. Johnson, R., Hoeller, J., Arendsen, A., Risberg, T., Sampaleanu, C. Professional Java Development with the Spring Framework. Wiley. 2005
23. Kim, L. The Official XMLSPY Handbook. Wiley Publishing. 2003
24. Koper, R.; Tatersall, C (Eds.). Learning Design. Springer. 2005
25. Krasner, G.E; Pope, T.S. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk 80 System. Journal of Object Oriented Programming 1(3), pp. 26-49. 1988
26. Moodle. More information available online at http://www.moodle.org (last visited on January 2006)
27. OKI Project. More information available ontline at http://www.okiproject.org/ (last visited January 2006)
28. OKI OSID Assessment specification. Available online at http://www.okiproject.org/specs/osid_2.html (last visited January 2006)
29. Remote Question Procotol Project. More information available online at http://mantis.york.ac.uk/moodle/course/view.php?id=14 (last visited January 2006)
30. Rehberg, S.; Ferguson, D.; McQuillan, J.; Eneman, S.; Stanton, L. The Ultimate WebCT Handbook, a Practical and Pedagogical Guide to WebCT 4.x. Ultimate Handbooks. 2004
31. Respondus. More information available online at http://www.respondus.com (last visited January 2006)
32. Richardson, C. POJOs in Action Developing Enterprise Applications with Lightweight Frameworks. Manning Publications & Co. 2005
33. SAKAI Project. More information available online at http://www.sakai.org (last visited on January 2006)
34. Sancho, P.; Manero, B.; Fernández-Manjón, B. Learning Objects Definition and Use in <e-aula>: Towards a Personalized Learning Experience. Edutech:Computer-Aided Design Meets Computer Aided Learning, pp 177-186. Kluwer Academic Publishers. 2004
35. Sierra,J.L.; Moreno Ger, P.; Martínez Ortiz, I.; López Moratalla, J.; Fernández-Manjón, B. Building Learning Management Systems Using IMS Standards: Architecture of a Manifest-driven Approach. ICWL 2005. Hong Kong, China. 31st July - 3rd August. Lecture Notes in Computer Science 3583. Springer. 2005
36. TOIA. More information available at http://www.toia.ac.uk/index.html (last visited January 2006)
37. Vogten, H.. CopperCore, an Open Source IMS Learning Design Engine. Available online at http://hdl.handle.net/1820/286 (last visited January 2006)
38. WebCT. More information available at http://www.webct.com (last visited January 2006)