

JISC DEVELOPMENT PROGRAMMES

Project Document Cover Sheet

Final Report

Project

Project Acronym	PyAssess	Project ID	
Project Title	PyAssess		
Start Date	1st April 2005	End Date	30th September 2005
Lead Institution	University of Cambridge		
Project Director	Steve Lay		
Project Manager & contact details	Steve Lay UCLES 1 Hills Road Cambridge CB1 2EU Email: swl10@cam.ac.uk		
Partner Institutions	n/a		
Project Web URL	http://pyassess.ucles-red.cam.ac.uk/ to be changed to http://pyassess.ucles.cam.ac.uk/ soon		
Programme Name (and number)	JISC E-Learning Programme		
Programme Manager	Tish Roberts		

Document

Document Title	<i>Final Report</i>		
Reporting Period	n/a		
Author(s) & project role	Steve Lay, Project Manager		
Date	2005-10-28	Filename	FinalReport-0a.doc
URL			
Access	<input type="checkbox"/> Project and JISC internal		<input type="checkbox"/> General dissemination

Document History

Version	Date	Comments
0a	2005-10-16	As requested in email 14th Sept 2005

Table of Contents

Table of Contents 2

Acknowledgements 3

Executive Summary 3

Background 3

Aims and Objectives 4

Methodology & Implementation 4

Outputs and Results 4

Outcomes 4

Conclusions 6

References 6

Appendix A - Package Overview 7

Appendix B - Relationship to RQP 11

Acknowledgements

The PyAssess project is supported by Cambridge Assessment and JISC. JISC has funded the project under the JISC E-Learning programme as a toolkit project under the E-Learning Framework and Tools strand. The author would like to thank Dr Robert Harding, Director of Innovation in Assessment and Learning at Cambridge Assessment, for his encouragement and support of the project.

Dr Graham Smith, as contractor to the project has continued to produce practical demonstrator software of immense value both to the PyAssess project and to the goal of achieving technical interoperability within the assessment community in general. Tom Pelly (Cambridge Assessment) brought his knowledge of Java to the experimental work of integrating the PyAssess source code using Jython. Alice Still gave valuable feedback during the development of QTI version 2.0 and started much of the development work of the Python modules from which PyAssess has grown.

The author would also like to thank Pierre Gorissen, SURF, for taking the original Python source code of the QTI migration project and, with no previous Python experience, turning it into a standalone executable with a simple graphical interface and distributing it via the SURF website. By making it accessible to a wider audience he demonstrated the value in developing the code further.

Finally, the author would like to thank Rowin Young and all the members of the CETIS Assessment SIG for listening to his presentations and providing the feedback which, perhaps more than any, has been used to steer the project.

Executive Summary

The PyAssess project has created a toolkit for implementing QTI-based assessment services using the Python programming language. The toolkit focuses particularly on response processing (a.k.a. marking). A command line demonstrator has been produced which illustrates the use of the toolkit and can be used for testing and debugging the operation of questions with complex response processing rules.

The use of the PyAssess toolkit from the Java language environment was investigated but the tools that enable this integration are not yet developed enough. The approach remains promising though, and the issues found appear to have been resolved by the tool developers so future releases may make this worth reinvestigating.

The PyAssess website is now being used to host a java-port of Dr Graham Smith's work. This should help to maintain the momentum behind publicly available open-source Java implementations of QTI.

The possibility of developing an RQP interface to the toolkit for providing marking services was investigated, however, a simpler model for interfacing with marking services is proposed to decouple it from rendering.

Background

The IMS QTI Specification is the key specification in the Assessment area of the E-Learning Framework. Version 2 of the specification was approved for publication on Friday, 4th February 2005. During the first phase of the specification development project a migration tool for converting items from version 1 to version 2 was created. The tool was developed using the Python programming language and was packaged as a binary executable file

suitable for use on Microsoft Windows and distributed through the SURF website.

It was soon clear that further work was needed to help bridge the divide between the version 1 and version 2 communities. In particular version 2 presents new opportunities for developers to provide assessment services using standard technical interfaces. Early implementations are required to help stimulate interest in using the new features and to provide feedback to the ongoing development of the QTI specification itself.

Aims and Objectives

The project set out to extend the source code of the existing migration tool and to use it as the basis of an open source toolkit for implementing QTI-version 2 based assessment services in Python within the context of the E-Learning Framework. The work also aimed to feed back into the QTI specification process, taking advantage of the Project Director's role as team-leader of the IMS Assessment Special Interest Group.

Methodology & Implementation

The project stuck fairly closely to the plan of ensuring test-driven development of the toolkit code. As such, rather than incremental releases a continual stream of working snapshots was created throughout the project, managed by the Subversion source code repository tool. The importance of these two methods, test-driven development and active use of source code control, was echoed by the speakers at the open source development session, JISC Conference (7-8th July 2005).

The proposed work was described at the CETIS Assessment SIG meeting (28th April) at the start of the project and mention was made of the project at CAA 2005 as part of Cambridge Assessment's sponsorship of that event. A demonstration of the final project outputs is being prepared for the forthcoming JISC Programme Meeting (15-16th November 2005).

Outputs and Results

The PyAssess project website maintains information about the project, including an RSS feed for notification of updates. It is the central point of distribution for the source code and tools generated by the project.

UML diagrams illustrating the way the PyAssess source is packaged and the way it relates to the provision of a marking service (or a hypothetical RQP service) are illustrated in the Appendix and are also available from the website as part of the source distribution.

The distribution itself contains a simple set of instructions for installing the modules into an existing Python installation. An overview of the code can be obtained using the pydoc tool that comes with Python - more extensive tutorial style documentation is not yet available.

A web-service enabled demonstrator that illustrates the marking of a limited number of items taken from a recent Cambridge Assessment development project (illustrating the automated marking of free-text responses) is available and will be demonstrated at the next JISC meeting (scheduled for Nov 2005).

Outcomes

A key outcome following directly from the successful bid for funding under this programme is the establishment of a web presence for the project to act as a focus for the distribution of the source code and tools produced. Equally significant, was the creation of the source

code repository to manage the source code and to make it available to selected developers in a form suitable for integrating changes into future distributions.

Throughout the project the emphasis has been on attempting to provide the depth of implementation required to support migration between versions of the specification. Also, helping people with their own interoperability testing by providing a fairly complete implementation to test their own content and tools against. This has meant grappling with some of the more esoteric issues involved in implementing QTI and a number of issues with the specification itself have been raised in time to be addressed by the IMS project team working on version 2.1.

Similarly, problems that exist outside of the specification being implemented can be flagged early with other developers (via the implementer's forum). To give one example, the implementation of the stringMatch operator in QTI requires a regular expression matching module capable of using the expression language defined by XML-schema. Such a module is not available in Python (or Java for that matter) and a parser for translating QTI regular expressions into Python regular expressions was required. This turns out to be a much harder job than might be expected as the standardization of regular expression syntax across language toolkits is poor - simply passing a regular expression to the underlying API is not generally enough. To make matters worse the regular expression language also has a syntax that is inconsistently documented in the XML schema specification itself.

A key concern at the planning stage of the project was whether or not the use of the Python programming language would enable the outputs to reach a wide enough audience. From looking at the discussion on the QTI developers' forum it seems likely that the majority of developers are interested in tools based on the Java language at this stage. Ultimately, a successful deployment of a web-service based tool need not require integration at the language level to be useful but, as distribution of the original migration tool illustrated, developers may wait until a packaged executable is available before actively working with a tool in a 'foreign' language.

Sadly, a version of the toolkit compiled into Java classes using Jython eluded the project. This was primarily because the version of the Python language implemented by Jython (a Python 2.1 compiler for Java) lagged too far behind the Python version used by the PyAssess source code (2.3). Some time was spent downgrading the PyAssess code to an earlier Python language version, but ultimately a near unsurmountable hurdle was reached with Jython's failure to differentiate ASCII and Unicode strings.

Jython is being actively developed and, at the time of writing, a new version that implements Python 2.2 is in alpha release which appears to have addressed this issue. Interest in the project remains high (though like many sourceforge hosted projects, it probably outstrips development activity), suggesting that the general approach of trying to compile Python source code into Java classes is still of some benefit and should be attempted when a suitable Jython release becomes available.

In the interim work has been started on updating the code previously created by Dr Graham Smith in Visual Basic. This code was previously made available through a website at Strathclyde University (alongside APIS) until that arrangement ended earlier this year. The lack of a free online demonstrator has hampered QTI awareness raising activities. By porting this code to Java it is hoped to make it useful as a source code distribution as well as reinstating the free demonstration on the PyAssess webserver. The resulting Java source code is complementary to that previously distributed by the APIS project and will hopefully provide a more easily accessible route for investigating QTI implementation issues with Java.

Following feedback from the CETIS Assessment SIG it was clear that a demonstration delivery engine was needed to help people gauge the level of implementation in the PyAssess toolkit. As a result, some time towards the end of the project has been spent creating a simple command line delivery system designed to help people debug their own (possibly hand-written) QTI example files while also allowing users to examine the internal state model of a QTI item session.

In creating the web-service enabled demonstrator it became clear that there was a fundamental difference between the intended usage of the existing RQP specification and the needs of a purely marking-based web-service. These differences are illustrated in Appendix B. The RQP specification actually states that "situations where only grading is required are likely to be fairly rare so the performance penalty incurred by needlessly rendering is not likely to be important for most systems." This presupposes that the system is capable of rendering at all. Furthermore, RQP is rooted in browser-based presentation models. There was never an intention to develop a full-blown QTI rendering system as part of the PyAssess project - this has been the subject of previous projects (such as APIS). Currently, a browser-based rendering component could be built using the PyAssess toolkit (which provides a suitable abstract interface for one) and used as the basis of an RQP server, but this has not been attempted. Instead, a simpler marking-orientated web-service has been developed using the SOAPpy web-service module for Python to illustrate the operation of a standalone response processor or marking service.

Conclusions

The transition of a community across a major revision of a specification, such as QTI v1.2 to v2.0 takes time. Tools are now emerging with v2.0 capabilities and demand for migration services is increasing. In addition to a tool for upgrading old content to v2.0 there is clearly demand for a tool to take v2 content and transform it into v1.2 content for use with existing tools.

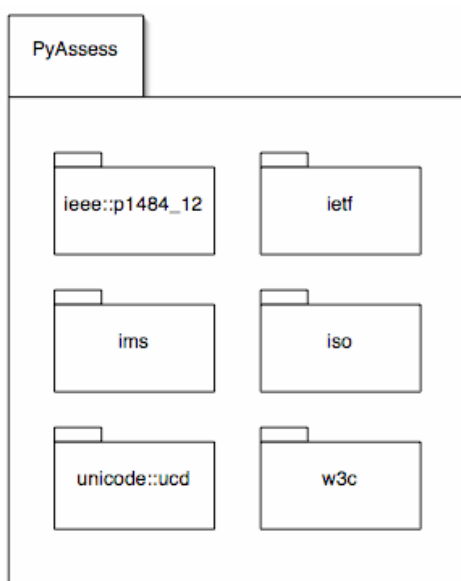
The RQP interface has pioneered the way in providing web-services to aid the delivery of assessment. If implementations of RQP can converge on a smaller set of item formats then supporting tools could become a useful way of packaging components suitable for building web applications. However there is also interest in other interfaces, such as those of pure marking services and those of complete assessment delivery systems (e.g., the interface being developed for communication between Learning Management Systems like Moodle and Assessment services as part of the IMS work on Tools Interoperability).

References

A list of further reading and associated web links is available from the PyAssess website, see the "Related Work" section: http://pyassess.ucles-red.cam.ac.uk/links/index_html.

Appendix A – Package Overview

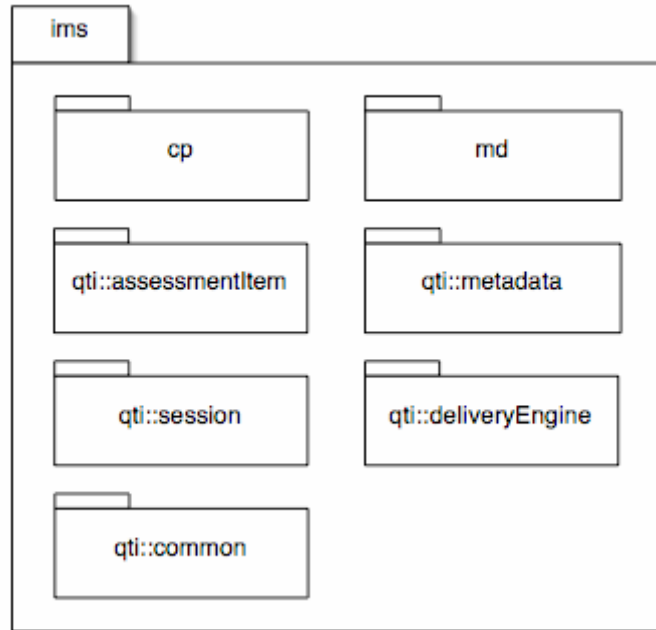
The PyAssess package consists of a number Python modules, arranged according to the groups responsible for publishing the various standards that are implemented.



The main purpose of PyAssess is to provide an implementation of the QTI specification. The sub-packages and classes that do this are in the `ims` package. The other packages provide supporting services as follows.

- The `ieee::p1484_12` sub-package provides an implementation of the IEEE LOM data model.
- The `ietf` package implements a number of utilities for dealing with supporting technical specifications published as RFC documents.
- The `iso` package contains a date/time module specifically suited to working with the formats described by ISO8601 and a module for working with language tags defined by ISO639.
- The `w3c` module contains utilities for dealing with concepts taken from XML, XMLNamespaces and XMLSchema that are reused within the IMS specifications.

The `ims` package contains Python modules for dealing with Content Packaging, and with the IMS LRM XML binding of the LOM model. The QTI sub-package contains classes for dealing with the `assessmentItems` themselves, for creating sessions (e.g., runtime data associated with a candidate's attempts) and a set of abstract classes suitable for building a delivery engine.



An example delivery engine is provided as part of the command-line demonstration application. An illustrative interaction with this application is provided below:

```
Welcome to PyAssess
```

```
This is the PyAssess demonstration program. This program is a command line QTI
delivery engine which allows you to validate, run (and debug!) your QTI version
2 questions.
```

```
Web-services enabled using SOAPpy
See http://pywebsvcs.sourceforge.net/ for more information
```

```
Type Help at the PyAssess> prompt for details of the available commands.
```

```
PyAssess> help
```

```
PyAssess Demo Commands
```

```
?
```

```
    same as help
```

```
ba
```

```
    same as beginAttempt
```

```
beginAttempt
```

```
    Start a new attempt at the current item
```

```
beginSession
```

```
    Start a new session with the currently loaded item
```

```
bs
```

```
    same as beginSession
```

```
debug on|off
```

```
    Turn on/off verbose error reporting (includes Python tracebacks)
```

```
endAttempt
```

```
    End the current attempt
```

```
help
```

```
    display this help text
```

```
load <file>
```

```
    load an assessment item from the given file
```

```
peek
```

```
    lists details of all the current session variables
```

```

quit          leave the demo immediately

setMarkingService <url>
              set the remote marking service to <url>

setms <url>
       same as setMarkingService

<num>
       During an attempt, select the control numbered <num>

PyAssess> load testdata/choice.xml
Read item id choice from testdata/choice.xml: OK
PyAssess> beginSession
PyAssess> peek
RESPONSE
      Type      : response variable
      Cardinality: single
      Base-type  : identifier
      Value     : None
SCORE
      Type      : outcome variable
      Cardinality: single
      Base-type  : integer
      Value     : 0
completionStatus
      Type      : built-in response variable
      Cardinality: single
      Base-type  : identifier
      Value     : 'not_attempted'
PyAssess> beginAttempt
Look at the text in the picture.

NEVER LEAVE LUGGAGE UNATTENDED

What does it say?

[1 ] You must stay with your luggage at all times.
[2 ] Do not let someone else look after your luggage.
[3 ] Remember your luggage when you leave.
PyAssess> 1
Look at the text in the picture.

NEVER LEAVE LUGGAGE UNATTENDED

What does it say?

[1*] You must stay with your luggage at all times.
[2 ] Do not let someone else look after your luggage.
[3 ] Remember your luggage when you leave.
PyAssess> peek
RESPONSE
      Type      : response variable
      Cardinality: single
      Base-type  : identifier
      Value     : u'ChoiceA'
SCORE
      Type      : outcome variable
      Cardinality: single
      Base-type  : integer
      Value     : 0
completionStatus
      Type      : built-in response variable
      Cardinality: single
      Base-type  : identifier
      Value     : 'unknown'
PyAssess> endAttempt
PyAssess> peek
RESPONSE
      Type      : response variable
      Cardinality: single
      Base-type  : identifier
      Value     : u'ChoiceA'
SCORE
      Type      : outcome variable
      Cardinality: single

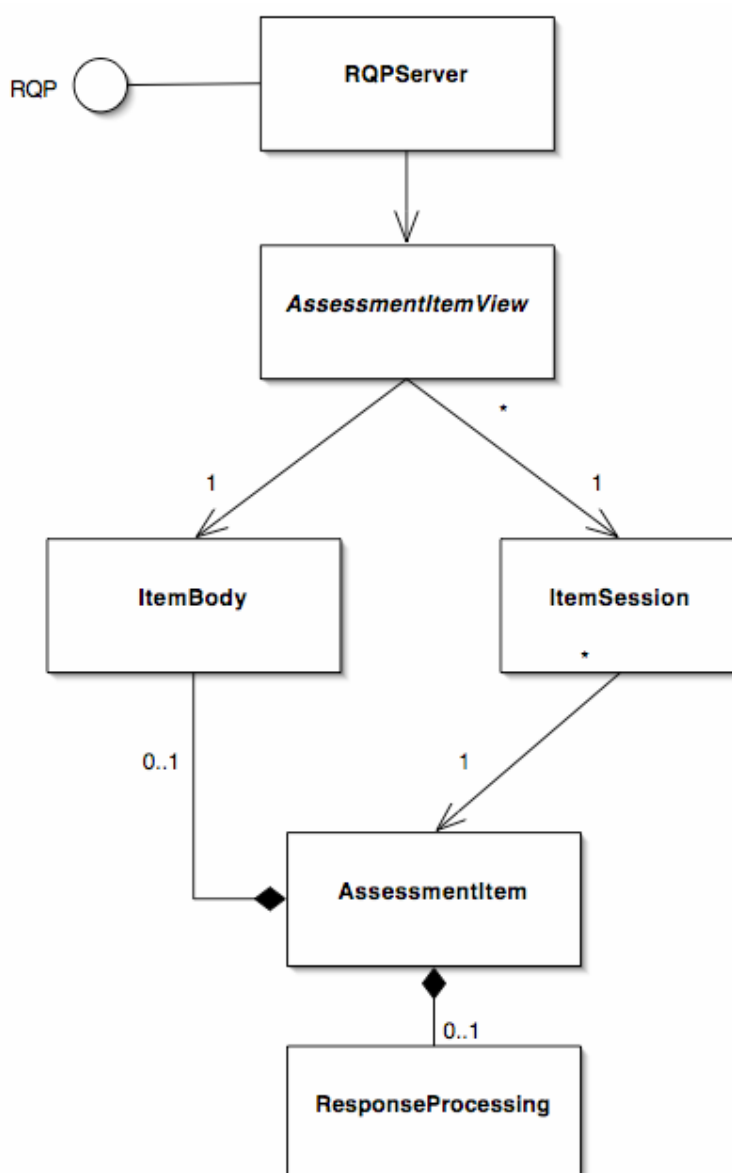
```

```
      Base-type : integer
      Value     : 1
completionStatus
  Type        : built-in response variable
  Cardinality: single
  Base-type   : identifier
  Value       : 'unknown'
PyAssess> quit
```

Appendix B – Relationship to RQP

The Remote Question Protocol (RQP) describes a web-service based interface that is implemented by an RQP Server. The interface supports two key features of an assessment delivery system: rendering and item-cloning. Rendering covers the transformation from an internal question format (e.g., QTI) and a rendered format (e.g., HTML fragments) with integrated management of the question session. In other words, the render method incorporates the process of marking or, in QTI terms, response processing.

PyAssess defines an abstract `AssessmentItemView` class for describing this type of rendering system. A rudimentary implementation of this class is provided in the package. This implementation is suitable for delivery through a command-line or terminal based system.

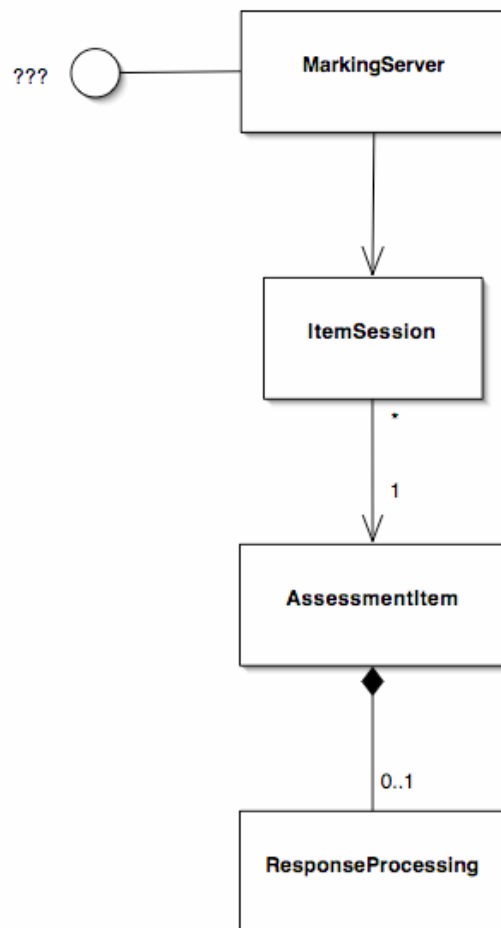


The RQP interface is very general – in particular, neither the question source input format nor the rendered output format are constrained by the protocol. Instead, the formats supported are advertised through the `ServerInformation` method and selected at runtime by the client. There are currently no known implementations that support QTI v2 format items.

Interest in RQP remains high though and future implementations that support QTI may become available.

The RQP specification states that "Situations where only grading is required are likely to be fairly rare so the performance penalty incurred by needlessly rendering is not likely to be important for most systems." In fact, the increased interest in sophisticated marking engines is likely to contradict this position over time with software designed specifically for marking and with no corresponding rendering capability. Furthermore, QTI allows the item body (for rendering) and/or the response-processing rules (for marking/grading) to be omitted when transferring data. This enables an important use case: the item can be handed to a delivery system without passing on restricted information about the mark-scheme. In this situation, the delivery engine is forced to seek an external marking solution before any feedback can be given to the candidate.

A simpler definition of a marking service, omitting the need for rendering, is therefore useful. As the authors of RQP point out, it is not the impact on performance that is a problem here, merely the difficulty of implementing an unnecessary rendering engine in a system that has been designed for, say, the batch processing of candidate responses.



The PyAssess package contains an experimental implementation of such a service. It is implemented in both server and client forms, the client being integrated into the demonstrator software. The interface itself is currently defined fairly crudely. In practice such an interface would be better served by communicating a result report in standardized format. The QTI project team are currently working on creating just such a format (to communicate the values of the student responses or other question variables) as part of the QTI v2.1 project.