Project no. 027087

TENCompetence

Building the European Network for Lifelong Competence Development

Project acronym: Integrated Project TENCompetence

Thematic Priority: 2.4.10

**D6.3 Learning activities and Units of Learning. Compilation of internal deliverable outcomes ID6.6 - 6.12**

Due date of deliverable: 31-05-2009
Actual submission date: 11-01-2010

Start date of project: 01-12-2005
Duration: 4 years

University of Bolton

Version 1.0

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Public</th>
<th>Restricted to other programme participants (including the Commission Services)</th>
<th>Restricted to a group specified by the consortium (including the Commission Services)</th>
<th>Confidential, only for members of the consortium (including the Commission Services)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Abstract (for dissemination)

The report describes the work of the TENCompetence project to provide a standards based open source infrastructure to support the implementation and running of learning and assessment activities. This deliverable describes the work done to further develop The TENCompetence Learning Design Toolkit (presented in D6.2) in the period January – July 2009. The principal components discussed include the ReCourse authoring tool with its associated plug-ins, further development and integration of the APIS runtime system for QTI, and work on integration of the LD Toolkit into the TENCompetence LifeRay system, and productisation of the TENCompetence Wookie Widget Server. A particularly significant strand of work has been the development of Astro, an entirely new Learning Design Player, which addresses a particularly weak area of IMS LD infrastructure.

Keywords List

IMS Learning Design Authoring, run-time IMS LD services, Astro, Units of Learning, ReCourse, Widget server

TENCompetence Project Coordination at: Open University of the Netherlands
Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands
Tel: +31 45 5762605
Table of Contents

1 Introduction ........................................................................................................................................................................... 2
    1.1 The Role of IMS Learning Design and QTI in TENCompetence .................................................................................. 2
    1.2 WP 6 tasks under DIP 4 and the work carried out ........................................................................................................ 4
    1.3 Delivery of code ............................................................................................................................................................... 6
2 LD Authoring .............................................................................................................................................................................. 7
    2.1 ReCourse 2.0 .......................................................................................................................................................................... 7
3 Task 2, Assessment and QTI support ........................................................................................................................................... 20
    3.1 The TENCompetence QTI Editor ........................................................................................................................................ 20
    3.2 NewAPIS QTI Runtime ......................................................................................................................................................... 25
    3.3 NewAPIS: Future tasks .......................................................................................................................................................... 28
    3.4 CopperCore/CCSI/SLeD/APIS framework ......................................................................................................................... 28
    3.5 Related publications ............................................................................................................................................................. 29
4 LD Runtime .................................................................................................................................................................................. 30
    4.1 The Astro Learning Design Player ..................................................................................................................................... 30
    4.2 Why a new Learning Design Player is necessary ............................................................................................................ 32
    4.3 Updated version of the Wookie Widget server and services ............................................................................................. 35
5 Integration ................................................................................................................................................................................... 46
    5.1 Integration of the components of the Learning Design Toolkit .......................................................................................... 46
    5.2 Integrating ReCourse with Fedora (Task Force 13) .............................................................................................................. 49
    5.3 Design of portlets for the integration of the LD Toolkit into the TENCompetence LifeRay based Personal Competence Manager .................................................................................................................. 50
6 Brief closing remarks .................................................................................................................................................................... 51
1 Introduction

The report describes the work of Work Package 6 of the TENCompetence project “Learning activities and units of learning”, between January and June 2009.

In this introduction we describe the role of IMS Learning Design and QTI in TENCompetence, providing the context for the work carried out and a summary of the strategy adopted by the project. We also set out the significance of the Learning Design Toolkit, and role that it plays within TENCompetence.

We then specify the tasks defined for WP 6 in DIP 4, and describe the work carried out. The description is kept brief, but links are provided to relevant project publications where appropriate. At the end of this section links are provided to the software produced in this period.

The work done is then described in greater detail in the main body of the report.

1.1 The Role of IMS Learning Design and QTI in TENCompetence

TENCompetence is committed to the use of interoperability specifications, and the systems it provides for the delivery of courses and assessments are based on IMS Learning Design (LD)\(^1\) and IMS Question and Test Interoperability (QTI)\(^2\). The tools provided by WP 6 support users in authoring, delivering and using learning activities and assessments which are compliant with these specifications. A learning activity within the context of the TENCompetence domain model\(^3\) may be very simple, for example a web page of information together with an instruction to read the text and text some notes. Alternatively it may be highly elaborated, for example with an extensive collection of study material with adaptive content and branching paths.

IMS LD enables the author to define a flow of learning activities called a Unit of Learning (UOL) which can be instantiated in runs for different cohorts of learners. The flow of activities is orchestrated by an engine, in our case (and almost universally for LD) Coppercore\(^4\). The user interacts with the UOL through a player application, which creates a user interface for the user, typically (though not necessarily) a browser. The IMS LD approach ensures that teachers can reuse the learning activities which they have carried out with one group of learners when they find need to repeat the course on a

---


\(^3\) http://hdl.handle.net/1820/649

future occasion. It also offers a very high degree of flexibility in the pedagogic structures which can be implemented.

In the first phase of the project it was foreseen that both simple and advanced activities provided by TENCompetence would be delivered using IMS LD, including both simple learning activities. However, this approach was modified once the capabilities of the TENCompetence Personal Competence Manager (PCM) became clearer. The PCM provides the environment in which learning activities are embedded and contextualised in terms of competences, and it itself has the ability to deliver simple learning activities (see Vogten 2008). The simplicity of these activities makes it easier for teachers to create them as and when they are required, and the support for reuse can be provided by a repository of learning materials, ideally with support for searching distributed web based resources (as is the case with the LearnWeb2.0 application available integrated into the PCM).

Because of this it makes little sense to require authors to use an additional layer of IMS LD tooling when there is no need to support reusability of learning flows, and when simple learning activities can be achieved more simply with the native tools of the PCM. Moreover, the creation of learning activities with the PCM has an additional advantage of flexibility, as its simple activities can be created and edited at any time, while IMS LD requires a distinction between design time and run time. As a result IMS LD is positioned within TENCompetence as an environment for the creation of sophisticated Units of Learning (UOLs), where the learning process to be orchestrated is sufficiently complex to require the use of the sophisticated capabilities of IMS LD in authoring, and where the time invested in preparation of UOLs makes reuse a significant issue. Examples include UOLs which require multiple roles, adaptive provision of resources, and complex flows of activity. The TENCompetence LD Toolkit provides a highly flexible environment with which collaborative and adaptive Units of Learning (UOLs) can be authored and run. Using the roles, properties and conditions available in IMS LD pedagogic approaches can be implemented which are beyond the capabilities of Virtual Learning Environments.

Despite these clear benefits, it must be accepted that the adoption of IMS LD has been much slower than had been hoped for. The reasons why technologies thrive or wither are complex, and we have discussed them in Griffiths and Liber “The prospects for Learning Design”. However, it is clear that one of the factors is that many of the applications available for making use of the specification have been no more than proof of concept tools, resulting from short research projects. The ambition of TENCompetence, and WP6 in particular, is to remedy this by creating a tool set which can be used as a delivery system in a real world context.

This has resulted in the first place in a focus on improving the quality, usability and functionality, of the tool set. Secondly we have resolved some of the key outstanding shortcomings of the infrastructure which were holding back adoption. Specifically,
a) IMS LD was created by adapting OUNL EML. In this process IMS stripped out all the assessment aspects of the specification so as to avoid overlap with existing specifications, principally QTI. However, practical use of the two specifications together has been very demanding, and a key aspect of WP 6 work has been to improve integration of IMS LD and QTI in authoring and runtime.

b) the integration of services in Units of Learning (for example, chat, forums, wiki, etc.), required intensive development work on each target platform, creating a significant overhead to adoption. The Wookie Widget server provides an effective solution to this problem. It also has applications in other areas of eLearning and beyond, and has attracted interest from Apache and W3C. As we point out in Griffiths and Liber (ibid.) the IMS LD specification was developed with the assumption that the landscape of eLearning would consist of a relatively large number of competing Virtual Learning Environments (VLE), and that the specification would provide interoperability between them. At present, however, we find that the sector is dominated by one very large commercial vendor, and a few Open Source systems, led by Moodle. At the same time the VLE is being questioned, as a historically necessary but transient type of application, for example in the Personal Learning Environment proposed by Liber\(^7\), which shares much of its approach with the Personal Competence Manager. In this context it is important to show how learning activities can be disaggregated from the VLE, and delivered in other systems. Consequently work is underway in WP6 to integrate the Learning Design Toolkit with the wider Personal Competence Manager system.

### 1.2 WP 6 tasks under DIP 4 and the work carried out

#### 1.2.1 WP 6 tasks under DIP 4

The activities of WP 6 in TENCompetence remain focused on developing and extending the infrastructure available in three task areas, which were inherited from DIP3. However, the focus of work in this final year of the project is on integration of the overall system, and so an additional Integration task was added under DIP 4. The tasks for WP6 are:

- Task 1: Authoring of Units of Learning
- Task 2: Assessment and QTI support
- Task 3: IMS LD Runtime
- Task 4: Integration.

The work described in this deliverable is all within this context of these four tasks. However, it is further articulated by the task force structure introduced in DIP 4.

As DIP 4 states,

> Many of the issues identified ... had to do with tools integration, and pertained to two or more Work Packages. The conclusion was that the present project structure was not well suited to address these types of issues, and that a matrix structure would be more suitable for this phase of the project.

---

\(^7\) See Interactive Learning Environments Special Issue: Narrative and Interactive Learning Environments, volume 16 Issue 3, 2008. Liber, Oleg & Johnson, Mark (Eds.)
This was realized by the formation of seventeen Task Forces, most of them addressing cross-WP issues. For each Task Force a provisional task(s)-outline was formulated, and a Task Force leader appointed. Each Task Force would formally fall under a Work Package. This restructured approach to development provided the framework for planning and carrying out the work reported in this deliverable.

The task forces positioned within WP 6 were as follows

a) Task Force 7 – QTI, responsible for integrating QTI tests into the Personal Competence Manager (situated in task 6.2)
b) Task Force 13 - ReCourse linking to Learning Objectives/Prerequisites and "Search Link to a UOL" (situated in task 6.1)c) Task Force 16 - Redesign SLED: Create a better, more stable, documented and modern version of SLED that is better integrated into the current set of TENC tools and services (situated in task 6.1)d) Task Force 18 - WYSIWYG XHTML editor for LD level B (situated in task 6.1).

1.2.2 Summary of the work carried out

This report covers the work carried out in the period January 2009 to July 2009. During this period the principal lines of work were

a) Development of a major new release of the ReCourse Learning Design Editor: ReCourse 2.0. This provides support for authoring IMS LD Units of Learning at levels B and C, and includes a QTI Editor plug-in (task 6.3, LD Authoring)

Recourse 1.0 provided tools which enabled learning designers and teachers with basic computing skills (e.g. the ability to create a webpage) to author flows of learning activities with any roles and activities which they chose to define, and to associate them with learning resources and services. Recourse 2.0 makes further advances in usability for these authors, particularly in the area of creation and management of resources. It also provides a visualiser shows the overall structure of the UoL, and an improved Checker for validating the UOL. Editors for Level B & C (primarily properties and conditions, and notification) can be used by authors with more technical skills to create automated scripted processes in UOLs, so teachers and technical experts can collaborate on authoring of complex UOLs.

b) Task Force 13 - ReCourse linking to Learning Objectives/Prerequisites and "Search Link to a UOL" Functionality has been built into the ReCourse Editor which enables Units of Learning(UOLs) to be packaged into zip files and uploaded directly to the TENCompetence Fedora data base from within ReCourse.

c) Further development of the Runtime System for QTI (task 6.2, Task Force 7) Work has been carried out to extend and enhance the rendering of question types provided by the APIS runtime system for QTI, to ensure smooth interoperability with the authoring components. A database has been implemented to provide persistence of user interactions.

d) Architectural and Design work on the Astro player (task 6.3, Task Force 16) An entirely new Learning Design player called Astro has been designed and the first stages of
development work carried out. This provides learners with a simple step-by-step view of the activities to be carried out (with an optional overview of the whole UOL), and a full set of flexible interface features delivered over the web. The long established SLeD “player” has also been extended and enhanced.

e) **Portlets for the integration of the LD Toolkit into the TENCompetence LifeRay based Personal Competence Manager.**

Design and initial implementation has been carried out on an LD Player Portlet (SLeD), an LD Administration portlet, and a link tool enabling learners to access learning activities on the LD Player portlet. These will be delivered in D6.4.

f) **Further development of the Wookie Widget server and new services for IMS LD. (task 6.3)**

To provide services which are both flexible and interoperable TENCompetence has developed the Wookie widget server based on the W3C widget specification. This delivers flexible services for LD runtime through web-deployed widgets with collaboration features, e.g. chat, forum, vote, and Google maps. In this period the code has been finalised, and new widget services provided.

In the following sections of this deliverable each of these areas of work and their results are described. There is then a section on conclusions and future work.

**1.3 Delivery of code**

The source code for the applications delivered is available under Open Source licenses on SourceForge, and as also stored permanently as zip files on the TENCompetence DSpace server.

The Learning Design Toolkit is delivered in two parts:

**a) Runtime**

The source code for the runtime system is available at [http://hdl.handle.net/1820/2278](http://hdl.handle.net/1820/2278)

This includes the components
- CopperCore LD Runtime Engine
- SLeD LD player
- newAPIS QTI engine
- Wookie Widget Server.

The latest build of the runtime system is made available at [http://www.tencompetence.org/ldruntime/](http://www.tencompetence.org/ldruntime/)

**b) Authoring**

The source code for the authoring component is available at [http://hdl.handle.net/1820/2247](http://hdl.handle.net/1820/2247)

This is the ReCourse Learning Design Editor 2.0, including the QTI Editor.

The latest build of the authoring system is made available at [http://www.tencompetence.org/ldauthor/](http://www.tencompetence.org/ldauthor/)
2 LD Authoring

2.1 ReCourse 2.0

Recourse 1.0 provided tools which enabled learning designers and teachers with basic computing skills (e.g. the ability to create a webpage) to author flows of learning activities with any roles and activities which they chose to define, and to associate them with learning resources and services. Authors can create the main structure of the units of learning from a single learning flow grid, and a simple but powerful system is provided to manage resources for inclusion in the URL. Nonexperts can create simple valid UoLs, either from scratch, or using a template. The development of ReCourse 1.0 and the lessons learned are discussed in a Journal article and chapter published by the team.

David Griffiths; Phillip Beauvoir; Oleg Liber; Mark Barrett-Baxendale. From Reload to ReCourse: learning from IMS Learning Design implementations. *Distance Education*, 1475-0198, Volume 30, Issue 2, 2009, Pages 201 – 222.


Recourse 2.0 makes further advances in usability for these authors, particularly in the area of creation and management of resources. It also provides a visualiser shows the overall structure of the UoL, and an improved Checker for validating the UoL. Editors for Level B & C (primarily properties and conditions, and notification) which can be used by authors with more technical skills to create automated scripted processes in UoLs. The fact that the system now meets the needs of both types of authors makes collaborative authoring of complex UoLs a much more practicable proposition.

2.1.1 Principal features of ReCourse 2.0

The ReCourse Learning Design Editor 1.0 was described and delivered in D6.2. This supported the creation of Units of Learning compliant with Learning Design level A. In the present D6.3 we deliver ReCourse 2.0, the TENCompetence tool for authoring Units of Learning compliant with IMS Learning Design (LD).

As described in D6.2, Recourse is an Eclipse Rich Client Platform application with a plug-in architecture, and this has been leveraged in the development of v2.0. The principal features of this release include:

- authoring support for IMS LD Level B, including interfaces for handling properties and conditions
- a visualiser which shows the overall structure of a Unit of Learning
- improved handling and authoring of resources.
- New Learning Object types have been added to the Environment Editor: Knowledge, Test and Tool.

There is also a component for authoring assessments and questionnaires using the IMS Question and Test Interoperability specification, which is described in the section of this deliverable dealing with assessment.
Editing Level B properties

"Level B" editing is provided in order to support the more advanced features of the Learning Design specification, namely "Properties" and "Conditions" (Level "C" adds Notifications to completion rules and Conditions). Properties can be created and edited in the "Properties" window accessed from the main "Window" menu (Figure 1). Properties can be added or deleted by right-clicking on the table or from the Add button in the top-right of the window. The Property is edited in the right-hand panel.

Figure 1: The ReCourse properties editor

References to Properties Level B Properties appear throughout the UOL, for example when specifying the completion rule for the UOL, a Module, Phase, or Learning Activity:
Figure 2: Setting the Completion Rule of a Learning Activity

**Advanced Property Values**

The value of a Property can be a simple number or string. However, it is possible to calculate the value of a Property by means of an expression. This can be achieved by clicking on the "Advanced" cell in the Property Value table (or by right-clicking). This invokes the Advanced Property Value Expression Editor dialog:
Figure 3: The Advanced Property Value Expression Editor

If a Property Value uses an advanced expression, then a tick is shown in the Property Value table.

**Editing Level B Conditions**

"Level B" editing is provided in order to support the more advanced features of the Learning Design specification, namely "Properties" and "Conditions". (Level "C" adds Notifications to completion rules and Conditions.)

Conditions can be edited in the "Conditions" window accessed from the main "Window" menu (Figure 4). Conditions and Condition collections can be edited by right-clicking on the tree or from the Add button in the top-right of the window. The Condition is edited in the right-hand panel. The Plus and Minus buttons are used to add and remove Conditional expressions.
Level C

Level C of the IMS LD specification enables the author to include notification elements in their Unit of Learning, enabling messages to be sent to systems and actors within the UOL itself, or beyond. Principally this occurs when an activity, act or play is completed.

The Level C Notifications elements surface at two points in ReCourse.

Firstly, the on-completion model is extended with a notification element. In the interface to the application this manifests as "When a UOL/Play/Act/Activity is Completed Send a Notification". This can be seen in the Inspector for a UOL/Play/Act/Activity under "Additional Completion Rules" (Figure 5).

Secondly, the "then" (and "else") model is extended with a notification element. In the interface to Recourse this manifests in the Conditions Editor as an action of "Then" or "Else" (Figure 6).
Figure 5: Editing additional completion rules
A significant challenge for authors of UOLs is to understand and keep in mind the overall structure of the UOLs which they are creating. The Visualiser of ReCourse provides support for authors in achieving this. Earlier visual representations of UOLs in authoring applications depend on graphical information which has been added to a custom file format in order to represent the structure of a UOL graphically (for example the recent Prolinx GLM (graphical learning modeller)). Rather it is generated by parsing the manifest of the UOL, and using the result to create a graphical representation. The result is that any IMS LD compliant Unit of Learning can be viewed in the Visualiser, whether or not it was created in ReCourse.

The "Visualiser" is a window that gives the author a "birds-eye" view of the currently selected Unit of Learning. It shows you the relationships between all of the LD entities. The current version display the LD entities at Level A - the Plays (Modules), the Acts (Phases), Role Parts, Activities and Environments.

The Visualiser is opened from the main "Window" menu. As with the Inspector view, the Properties view and the Conditions view, this is a contextual window which means that if you have more than one
Learning Design open, it will display the visualisation of the LD that has the focus. The user can navigate through a Unit of Learning by clicking on a node to go deeper into the structure. There is a forward/back button on the toolbar which provides access to the history of locations shown in the Visualiser.

The Visualiser is shown in Figure 7.

![The ReCourse visualiser](image)

**Figure 7: The ReCourse visualiser**

The user has a high degree of control over the presentation of the visualisation.
- The “Explanation” window at the top of the graph can be hidden or shown
- Zoom in or out from the menu button (or right-click)
- Drag the nodes to rearrange them
- "Pin" selected nodes and un-pin them – useful in highlighting child nodes which can be dragged to another position
- Ctrl-clicking of nodes allows multi-selection
- The default layout type (horizontal or vertical) can be selected from the menu bar.
Figure 8: Choosing the Visualiser layout

Authors can return to the original graph by using “re-layout” from the "Layout" button (or right-click).

2.1.2 Usability enhancements

In addition to major new features (support for levels B and C, Visualiser...) there have been numerous other enhancements. These are documented in the release notes REF, but the following are worthy of note here.

Enhancements to the Checker

The Checker enables the author to ensure that the Unit of Learning which they have produced complies with the IMS LD specification, before they package it to a zip file. Its functionality has been greatly enhanced in this release, including the following features:

- A frequent problem reported by authors was that they forgot to declare the level of their UOL (A, B, or C). In this release the checker not only parses the UOL to see if level B or C has been used, it also automatically sets the level declaration to the correct value when saving or running the Checker
- Double-clicking on an error in the LD Checker opens the corresponding editor tab or Inspector for the object
- A new Warning type has been added to the LD Checker for issues that pass XML validation but which are not strictly correct
- SAXParser XML validation has been added to the Checker and Package/Publish/Upload dialogs.
Opening and Saving

The process of opening and saving UOLs was greatly enhanced in version 1 of ReCourse, by enabling it to open and save UOLs without the need for an intermediate file format (used by all previous high level editors). The process has been enhanced by checking the file format validity on the Open action, so that the user can now only open files named "imsmanifest.xml".

Moreover, on packaging or uploading a UOL, the author is warned if they have made unsaved changes, and offered the option of saving before they create a content package suitable for uploading to a server.

2.1.3 Authoring content

Authoring and managing resources

A Unit of Learning is composed of many different files, much like a website. The specification states that authoring should not be carried out with references to the file names of the physical files, but rather to higher level resources which are named by the author and associated with physical files inside the UOL manifest. This approach to managing content provides flexibility and aids maintenance of the UOL, but also adds complexity to the authoring task.

ReCourse 2.0 hides this complexity from the author. For example, a new activity can be created directly in the main learning flow grid (Figure 10).
In previous versions of ReCourse the author had to enter into the content management system, and select a resource to display in the activity, having previously created or imported a file. In ReCourse 2.0, the author can simply double click the activity, and start editing it with the built in editor, or with a linked HTML editor. The application manages the creation of a file in the UOL folder, the creation of a resource pointing at that file, and the association of that file with the activity.

Similarly, a learning object can be authored in context (i.e. inside an environment”) and ReCourse manages the creation of the files and their relationship to resources.

The result is that authors can create a Unit of Learning and never need to be aware what files are being created, and how they are being linked to the activities and learning objects which they have authored. Nevertheless, the full content management system is available to support more advanced authoring, import of resources, and maintenance of the UOL.
Authoring XHTML content

Many of the files which make up a Unit of Learning are simple HTML or XHTML files. However it is also possible to make use of XHTML files which include properties defined within the Unit of Learning, as we now describe.

In IMS LD there are a family of properties (personal, local, global...) which are place holders for values. You can set them and read them. So you into your property you can put a string (e.g. a piece of text), an integer, or whatever. You can use it to keep track of something that is going on in your UOL, for example comments submitted by users, number of times that a user has uploaded documents, scores in tests, etc.

Properties are mainly used in two ways:

a) In conditions, which expressions which are typically "if / then", for example "finish this activity when half of the learners have uploaded their assignments", or "if the learner's score is more than 60% as an average of the last five activities, then assign them this activity". The conditions are typically conditional upon calculations using properties defined in the UOL.

b) Embedded in the pages shown to the user. Pages of this sort are known as “IMS LD Content”. In the following examples the text between *...* is a paraphrase of what the XHTML will achieve:

- using the value of a property in a page to be shown to the user, e.g. "Congratulations, your score is *insert value of property learnerScore here* and your teacher has provided the following feedback *insert value of property teacherfeedback here*"
- providing ways in which the user can input information, e.g. "Please write a paragraph describing the most useful aspect of this course *here provide a field to gather text and put it into the property myusefulaspect*"

All this process can be handled directly in ReCourse, with the exception of the creation of the properties into the XHTML pages. We now discuss why this is the case, and the support which has been provided for authors in ReCourse.

WYSIWYG XHTML editing per se is not problematic, and indeed the bundled Rich Text Editor Plug-in in ReCourse generates XHTML. The challenge is to support the author in inserting IMS LD properties into the content. The option of building on the existing Rich Text Editor (not developed by TENCompetence) was explored. However

a) this plug-in does not work on Linux based systems
b) the quality of the editor is not adequate for high end users.

No satisfactory free Open Source and cross platform external editor could be identified. Moreover the task of creating IMS LD content involves generic XML authoring, a task which is largely carried out by technical experts. Experience with IMS LD and other specifications (e.g. SCORM) indicates that these advanced users tend to employ their own preferred XML authoring tool for this purpose.

Consequently the following functionality was implemented in ReCourse 2.0:

1) Authors can choose whether to use the Rich Text Editor plug-in, or to designate their preferred external editor.

2) A “smart copy” functionality is provided on properties in ReCourse. This means that when the author right clicks on a property they can select from a drop down menu to indicate how they want...
to use it. This then places in the clipboard the string that is needed to paste into the XHTML document.

3) New Resource, New Item, and New Activity wizards have option to create HTML ("webcontent") or XHTML ("imsldcontent") files.

4) Resources are set to "imsldcontent" type by default if they have an .xml or .xhtml extension.

**Refactoring to support reuse of ReCourse code**

ReCourse has been designed so that it will be easy for others to build on it. In response to requests from other developers it became clear that this could be enhanced by using a standalone Java library to model IMS-LD and create the manifest which could be re-used in other applications. In response to these requests the IMS LD Model Java classes in ReCourse were refactored into a standalone Java library module with no dependencies on Eclipse or ReCourse, the only remaining dependency being on the JDOM library, used to model the XML. The code is published separately on the TENCompetence SourceForge CVS repository as

```
Host: tencompetence.cvs.sourceforge.net
Repository path: /cvsroot/tencompetence
Connection type: pserver
User: anonymous
```

The modules needed are:

```
org.tencompetence.jdom
wp6/org.tencompetence.imsldmodel
```

These are set up as Eclipse plug-ins (bundles) so they can be used as Eclipse projects, but they can also be used as plain Java libraries. In the root package there is a class, *Example.java*, that gives example code of how to create a new LD Model, read it in from an imsmanifest.xml file, and then save it again. However, the best examples of usage are in the ReCourse Editor code, in the main module, org.tencompetence.ldauthor, which has been adapted to make use of the independent library.
3 Task 2, Assessment and QTI support

Work on this task has been articulated and managed by Task Force 7, coordinated by partner UPF. This task force has been working on the development and integration of applications which work with the Question and Test Interoperability specification published by IMS\(^8\), and covering both authoring and runtime components. The key interventions have been to further develop the APIS runtime system\(^9\), to develop a QTI editor, and to fit these for integration with the LD authoring and runtime systems and the Personal Competence Manager produced by TENCompetence.

3.1 The TENCompetence QTI Editor

A QTI editor was developed and reported in D6.2. This first version supported the creation of the following questionItems: Multiple Choice, Multiple Response, Yes or No, Likert, Open Question, Fill in the blank, InlineChoice. In the period covered by D6.3 the functions of QTI editor have been extended to include six additional types of questions: match, associate, gap-match, hot-text, order, and slider. These are compliant with IMS QTI 2.1, but do not implement the whole of this extremely extensive specification.

When a user selects one of these seven types of questions, s/he provides the text of the question. In most cases answers are entered (with the correct one identified) and the associated score and feedback. Finally if the user is satisfied s/he can add the question in a test. When the user has created all the questions, s/he has to fill the final “Report Results” of the test. In this option s/he can select which information will be showed to the student (e.g. correct responses, responded responses…) and s/he can add a test-feedback which will depends of the percentage of correct responses. The tests/questions are saved as an IMS QTI compliant file.

The main edit functions of the question types added in this release are described from the perspective of the author.

---

\(^8\) http://www.imsglobal.org/question/

3.1.1 Authoring a match question

A match question presents candidates with two sets of choices (called simple-match-set) and allows them to create associations between pairs of choices in the two sets. Associations cannot be made between items in the same set. As shown in Figure 12, the user can describe the question text with a text editor and define and order two sets of choices: source match set and target match set. Associations are made by clicking on radio buttons, and a maximum number of attempts by the learner can be set, as well as the score for each one. At runtime, a response variable is generated automatically for each match, in response to the user’s answer, which is used to keep track of the user's performance in the test, through the variable (id="SCORE").

![Figure 12. The user interface for editing a match question](image)

3.1.2 Authoring an associate question

An Associate question is similar to a match question, because an association between two choices must be defined. The difference is that the two choices are selected from the same set. The author provides a question text and a set of choices. They choose a source and a target, and then define an association
with a restriction on the max-match of the choices.

### 3.1.3 Authoring a gap-match question

A gap-match question contains a number of gaps defined in the question text and a set of choices. The candidate will fill in a gap with a choice. As shown in Figure 4, the user interface to edit a gap-match question consists of three panels: a panel for defining question text and a set of gaps, a gap match list, and a number of choices. The user creates the question text and then clicks the “Edit” button to start the definition of gaps. The user can select a word or a phrase by marking it as a selection (for example “summer” in the figure below). A gap is created by clicking the “Create Gap” button. The gap text is used to generate a choice, which will be added at the end of the choice list.

If the text of a newly created gap is the same as the text of an existing choice, the editor will not generate a new choice. The created gap and the existing choice will be associated and listed in the gap match list as a new row. In addition, the user can manually define new choices, which will not be associated with any gap. When the “Reset” button is clicked, all defined gaps, choices, and associates are removed.

![Figure 13: The user interface for editing a gap-match question](image)

TENCompetence – IST-2005-027087 22/51
3.1.4 Authoring a hot-text question

A hot-text question presents a set of choices to the candidate represented as selectable sections of text embedded within the question text. The author can create question texts and define hot-text within the question text, in the same way that gaps are defined in the previous question type. The defined hot-texts will be presented in the hot-text list. A maximum number of correct choices can be defined. If this is “1”, the learner will use a radio button to indicate the correct answer. If the maximum is defined as “0” (i.e. no restriction) or a number which is larger than 1, the user then can select a number of choices (not larger than the maximum number) by clicking corresponding check buttons. In this case, the author can define a score for each choice.

3.1.5 Authoring an order question

An order question provides a number of choices, and the learner has to reorder them. The author uses the QTI editor to define a question text and a list of choices. In addition, the user can change the order of the choices and define the correct order. The shuffle button can be selected if the sequence of the choices should be rendered randomly when they will be presented the candidates.

3.1.6 Authoring a slider question

A slider question presents the candidate with a control for selecting a numerical value between a lower and upper bound. As shown in Figure 5, the QTI editor enables to define question text and a scale with a lower bound, an upper bound, and a step. The user can decide whether or not each step on the slider should also be labeled and choose a vertical or horizontal slider. A numerical value can be selected by dragging/clicking the handler in the scale, together with scores. A correct selection can be defined by clicking a radio button.
Figure 14. The user interface for editing a slider question

Finally this QTI questionnaire is added as a resource in an activity into the flow of learning activities designed with ReCourse.
3.2 NewAPIS QTI Runtime

NewAPIS\(^{10}\) is an engine which interprets QTI tests and questionnaires, and it is an extension of an existing Open Source engine called APIS. NewAPIS has been improved to support the functionalities implemented in TENCompetence LD Toolkit. The current version of this engine renders and handles interaction with the seven questionItems created by the editor. The questions can be answered in any order that the student wants. When the student answer a question, newAPIS checks it and provides modal-feedback (if this has been authored by the teacher beforehand). When the test is finished, the engine computes all the scores and shows the report result and the test-feedback. The questionnaires can be published and visualized in the TENCompetence LinkTool (see Figure 2).

![Figure 15.: Visualization of a QTI-questionnaire](image)

To enable the communication between LD activities and the QTI questionnaire a set of LD services for Coppercore service integration has been developed.

3.2.1 Rendering Inline Choice

The inline choice item has been fully implemented in NewAPIS (see Figure 9). This item involves the user in providing more than one piece of input (referred to as having a cardinality of more than 1). NewAPIS has to interpret these inputs in order to compute the score.

---

Feedback elements

3.2.2 Modal Feedback

Implementation of the Modal feedback element makes it possible to show a feedback message for each question rather than wait till the end of the test (see Figure 17).

Figure 17: Visualisation of the modalFeedback element

3.2.3 Test Feedback

As the name suggests, test feedback is given at the end of the test. It shows the message defined by the designer of the test during authoring (see Figure 18).
3.2.4 Test rendering

Order in tests

QuestionItems can be answered in any order in a test. This means that item composition in tests-parts is fully implemented.

Overall workflow

With the persistence provided by the database added in this release, the test can be left in the middle of response and the user can finalize it later on. Once the test is complete, it can be repeated again. This workflow has been designed for the specific needs of the TENCompetence pilots.

Test feedback only at the end of the process

Test feedback now appears only at the end of the test (when all items have been responded). In previous releases it was always shown.

3.2.5 Persistence of user interactions

In this release, a first implementation of a data base has been developed in order to save the data-information of the interactions that the users perform in the questionnaires. This was initially implemented using MySQL, by creating a database called ‘sledapis’.

However, the TENCompetence version of the CopperCore Runtime Environment (CCRT) also allows for the use of a Hypersonic database, which can be made to run automatically (using MBeans) once the JBoss server within CCRT is started. This simplifies the installation process.

This use of MBeans followed existing practice in the TENCompetence CCRT, as the CopperCore ‘publication’ database and the Widget servers ‘widgetdb’ database were implemented in this way. However, the existing MBean service which started these databases using Hypersonic, was not very generic. It originally started these on two separate ports for example and adding new databases to it
meant you had to update and recompile the MBean code. Consequently changes were made to the existing MBean so that it could run more generically. The MBean now looks for a properties file (server.properties) at the root of the application, once it initiates. In this file we can define which databases to automatically start and where each databases resources are located.

Some Dao classes have been added in order to save information of the inputs of the test in a Data Base (DB). The stored data is reachable via a SQL client. This DB (given a user and a run) saves: the score obtained for each answer, the text of the open question answers and the number of times the user has done this test. The feedback information is also stored into the DB. To distinguish between different users, the primary key of answers tables is the users name and run id. A new version of this data base is under development (see future work).

3.3 NewAPIS: Future tasks

3.3.1 Extension of the data base

Proposed enhancements for the database are to implement a more relational design and to collecting more information. Several models have been considered and finally we have selected one which avoids duplication of data that is already available in the XML files which make up the Uol. This new persistent data would make it possible to provide the teacher with statistical information of the results in a tool other than APIS.

3.3.2 IMS LD and IMS QTI interoperability

Work is still required to improve the ability of newAPIS to interact with UoLs at level B, where the results of a QTI test set a property in LD. This can be used to create UOLs where a QTI result can be used to create an adaptive flow of LD activities, or to provide feedback to learners on their progress.

There are two levels at which this interoperability can be addressed:

a) at the level of the test result, accumulating the results of all the test items.
b) at the level of the individual questions (enabling finer grained adaptivity to be carried out).

Both these will be addressed in the coming period of work, with priority given to a).

3.4 CopperCore/CCSI/SLeD/APIS framework

3.4.1 Changes made to the Server Environment

The latest version of newAPIS required the use of a database to store information. This was initially implemented using MySQL, by creating a database called 'sledapis'. However, the TENCompetence version of the CopperCore Runtime Environment (CCRT) also allows the use of a Hypersonic database. The rationale behind this is that Hypersonic can be made to run automatically, once the JBoss server within CCRT is started. This is good for first time users who subsequently don’t have to configure a database in order to use the software.
JBoss uses something called ‘MBeans’ which allow you to define services to start, during the boot sequence. For example one such service could be a database service. There were already the CopperCore ‘publication’ database and the Widget servers ‘widgetdb’ database in existence within the framework. However, the existing MBean service which started these databases using Hypersonic, was not very generic. It originally started these on two separate ports for example and adding new databases to it meant you had to update and recompile the MBean code.

Changes were made to the existing MBean so that it could run more generically. The MBean now looks for a properties file (server.properties) at the root of the application, once it initiates. In this file we can define which databases to automatically start and where each databases resources are located.

The priority tasks for work with APIS QTI in the final project period (to be reported in D 6.4) are:
- Implement a newAPIS version which has to work without Coppercore (IMS LD) in order to interpret QTI test independently, to be used with the Personal Competence Manager.
- Continuing the work of improving some aspects of newAPIS: persistence of the data, visualization and interaction of the tests.
- Solving the codification problem with accents in QTI Editor.

3.5 Related publications

The following publications discuss aspects of TENCompetence related to assessment and QTI.


4 LD Runtime

4.1 The Astro Learning Design Player

4.1.1 Existing approaches to representing Units of Learning at runtime

IMS Learning Design Units of Learning (UOLs) have the following hierarchy…

```
UOL
|   (omitting the METHOD element for clarity)
|   -- PLAY (uols can contain many plays)
|   |   --- ACT (plays can contain many acts)
|   |   |   -- ACTIVITIES (acts contains activities and structures of activities)
|   |   |   |   -- ENVIRONMENTS (activities can link to an environment)
|   |   |   |   |   -- SERVICES (environments contain services)
```

The most obvious way to show this hierarchical structure is as a tree, and existing players have tended to transform the XML into a tree representation. In this they follow the precedent established by the player bundled with the CopperCore Learning Design Engine.

![Image of the original CopperCore Learning Design Player](image_url)

Figure 19: The original CopperCore Learning Design Player
This approach has the merit of showing everything in the navigation tree. However, it can make the navigation user interface cluttered and unintuitive. This is particularly true when the tree is expanded, as the structure starts to become complex, with lots of scrollbars.

The SLeD player (which has been adapted and enhanced for use in TENCompetence) maintained the tree for navigation, but tried to simplify this by hiding the UOL, PLAY and ACT levels (see figure 20).

![Figure 20: The SLeD Learning Design Player](image)

The simplifications introduced by SLeD help to reduce complexity, but in doing so they introduced some problems.

a) A UOL has resources attached to it, for example prerequisites, learning objectives. These can’t be accessed in the User Interface.

b) Once the UOL is completed, you cannot see in the UI that has been finished.

c) A UOL can have feedback to show the user when everything is finished – there is no way of accessing it.

d) PLAYS and ACTS also have completion/feedback states which are missing from the UI.
4.2 Why a new Learning Design Player is necessary

Potential solutions to the design of interfaces for IMS LD have been greatly constrained by the technology employed. The CopperCore and SLeD players use an approach to Web programming which precludes the use of floating windows, widgets and Ajax methods which enabled web application interfaces to make great strides forward in recent years.

Moreover, experience of enhancing and optimising the SLeD server for use in TENCompetence pilots has shown that the code base is not sufficiently well structured to be a long term solution for delivering UOLs to users. This is not surprising when one considers that SLeD was originally developed in a six month project funded project (funded by JISC in the UK) whose aim was not to create a delivery system, but rather “to evaluate whether Learning Design works as a methodology which can be used in practice to support online course presentations”.

Since delivery of SLeD in 2004 it has been extensively patched to meet the needs of demonstration and piloting in other projects. The experience of doing this in TENCompetence made it clear that it was not a solid basis for development, even if the technological limitations of an outdated approach to Web programming could be overcome.

Consequently the decision was taken to build a new player application which would interpret and represent the output of the CopperCore Learning Design Engine, and this decision was reflected in DIP4.

The goal in the Astro player is to move towards a UI which can hide everything but the ACTIVITIES from the user, but can achieve this without depending on TREES. To achieve this the UOL, PLAY and ACTS need to be treated as entities in the model, and the challenge of the interface design is to provide an elegant and effective way for the user to access the resources which are attached to those nodes.

To further inform the design, an additional session was held where existing LD runtime systems were analysed and issues identified. For example "When using SLeD a user does not easily see where they are in the current UOL. It would be nice to have the ability to have activities marked in sequence i.e. 'You are on activity 2 of 6.

The representation of Units of Learning in Astro

Detailed design work on Astro was commenced with an intensive working meeting on 17th – 18th March. This included four members of the development team, plus two teachers (also expert in eLearning) from Manchester Metropolitan University, who were unconnected with the TENCompetence project, in the role of both co-designers and critics. The meeting included an overview of learning design solutions developed to date (SLED, LAMS, MOT+ etc), and presentation of the initial wireframes proposed by the developers. The main activities of the meeting consisted of working in two teams to produce paper prototypes of the Astro interface. The process was facilitated by Scott Wilson of the University of Bolton, and consisted of two phases.

The representation of Units of Learning in Astro

Detailed design work on Astro was commenced with an intensive working meeting on 17th – 18th
March. This included four members of the development team, plus two teachers (also expert in eLearning) from Manchester Metropolitan University, who were unconnected with the TENCompetence project, in the role of both co-designers and critics. The meeting included an overview of learning design solutions developed to date (SLED, LAMS, MOT+ etc), and presentation of the initial wireframes proposed by the developers. The main activities of the meeting consisted of working in two teams to produce paper prototypes of the Astro interface. The process was facilitated by Scott Wilson of the University of Bolton, and consisted of two phases.

**Phase 1. Brainstorm of various aspects of the design**

- Personas
- Service design: how LD Player fits into the user's world (standalone embedded... what happens before and after use etc)
- Motivation (problems & ideas)
- Metaphors
- Scenarios
- Models (user’s model of the system)
- Tasks and flows
- Displays
- Controls (buttons and handles).

Phase 2: Development of paper prototypes in two teams, and presentation.

The designs produced were passed on to the lead developer, who used them to inform the development of the mock up interface presented below.

In the Astro user interface it is proposed to move away from division of the screen into the tree view for navigation (left hand side) and the content page (right hand side), although the tree view may be maintained as an option. The aim is to make it possible for everything in the User Interface to be contained within collapsible widgets. A Menu widget shows ACTIVITIES (although some will higher level elements than activities as they are defined in IMS LD). This can view and interact with these activities as a filmstrip, navigating forwards and backwards using the arrows. When a user clicks on an ACTIVITY several widgets will appear.

- A description widget
- A prerequisite widget
- Learning Objectives widget
- Widgets for any services found in attached environment of the ACTIVITY.

The filmstrip and widgets can be seen in Figure 21 below.
4.2.1 Dealing with the consequences of a focus on activities

An activity-centric user interface creates problems with the presentation of certain types of information. These are similar in some aspects to those which were identified for SLeD above, but fortunately Astro has a much richer set of interface elements with which the issues can be addressed. The problem may be summarised as follows: Astro hides the UOL, PLAY and ACT elements in the UI, but there are times when items associated with those elements need to be displayed to the learner. Specifically:

1. How do we show the user the feedback which they should be given once an activity or an act has been completed? ACTIVITY level feedback could be present using a designated feedback widget for the currently selected ACTIVITY. (before completion of the activity it would say something like “look here for feedback once you have finished”) Alternatively (and perhaps more elegantly), another activity entity(box) could be dynamically generated in the filmstrip (i.e. the next thing to do) which would be something similar to “look at the tutors feedback” (if there is no feedback, then nothing is shown). A similar dynamically generated activity entity(box) could provide the user interface for learner feedback provided at the level of an ACT.

2. How should plays be represented? Plays present particular problems, because they can run in parallel with each other. Consequently a mechanism is required in the user interface for changing between
plays. For clarity the learner should always be clear about which play they find themselves in, and so we have an added aim of avoiding showing all of them all at once (as other players do). The proposed solution is a drop down combo box, containing PLAYS. The user can select one of the plays from the list and the rest of the UI updates. It is not yet clear if this will be presented to the user in the filmstrip, or in a menu of the main window. Similar considerations apply to feedback on completion of a Play as were discussed in point 1 above in relation to activities and acts.

3. How should elements at the UOL level be presented? The prerequisites and learning objectives should be made available from the start of the LD. Certain outcomes can generate the completion of the LD and make feedback available, which should be presented at the appropriate time. Prerequisites and learning objectives could be presented as the first box in the filmstrip, which would launch a widget designed to represent UOL level items. A similar solution could be provided for feedback at the end of the UOL, which reopens the same widget, this time with UOL feedback. Alternatively the information could be presented as a menu on the main window.

4.2.2 Progress to date on Astro

Following the initial design stage an architecture for the Astro player was developed (to be submitted together with the player in D6.4).
The candidate technologies were explored by creating initial mock ups. The principal candidates were Open Lazlo\(^{11}\), Flash, and a combination of JQuery and Direct Web Remoting (the latter is also used in wookie). The decision was made to use the last of these, which had already been successfully applied in the TENCompetence Wookie Widget server. At the time of this deliverable parts of the Astro user interface have been constructed (for example the filmstrip), and have been submitted to the TENCompetence CVS. However it is still not developed enough to actually play UOLs at that time. In the coming period the interface will be fully developed and connected to the CopperCore engine.

4.3 Updated version of the Wokie Widget server and services

4.3.1 Overview

The solution to the provision of runtime services for IMS-LD developed by the project was centred on the use of a Widget Server, integrated into the existing IMS-LD infrastructure, as proposed in (Wilson, Sharples, & Griffiths, 2007).

Widget type technology is currently in wide use and across most platforms in one shape or another. Examples of this technology include Apple Dashboard\(^{12}\) widgets and the gadgets found under Microsoft Vista Sidebar\(^{13}\). For our purposes, a widget is a small software component, which can be configured to work within the context of a learning design.

A Widget Server was developed for TENCompetence and given the name “Wookie”. It is responsible

---

\(^{11}\) http://www.openlaszlo.org/

\(^{12}\) For information about Apple widgets see http://www.apple.com/macosx/features/dashboard/.

\(^{13}\) For information about Vista Sidebar see: http://www.microsoft.com/windows/products/windowsvista/features/details/sidebargadgets.mspx
for providing a particular widget that the IMS Learning Design runtime system might request, so that it can be presented to the user within the Unit of Learning. As far as the user is concerned, the widget servers are completely integrated into the Learning Design Player. This work was completed in May 2009.

### 4.3.2 Development Activity

Up until July, work progressed on the Widget Server under WP6, and included the completion of the core code, as well as sample plugins and widgets.

Key development tasks for this period included refactoring of the service to be much more generic and usable outside of a Learning Design-specific environment, improving the API for integration of the service. This makes it much more likely that a community of Open Source developers will in future maintain the server and create an ever richer set of widget services. This approach has led to interest from the Apache community, and discussions are progressing on submission of Wookie to the Apache incubator. Another strand of work has focused on improving conformance to the W3C Widgets specification.

In the earlier releases there was a single API for all aspects of integration, and in later releases this was refactored along using REST design principles into a set of services for Widgets, WidgetInstances, Participants, and Properties. This created a more intuitive pattern for developers, who can then use standard http verbs to interact with the service; e.g. to get a list of widgets they call GET /wookie/widgets and to create an instance they send a POST to /wookie/widgetinstances. This new API was implemented in existing plugins.

Additionally there was a requirement from the LAMS development team to include a mechanism where a designer could instantiate a widget within the authoring environment (e.g. to set properties such as the question in a quiz) and then “clone” the widget instance to use it for a particular run. As this is valuable functionality within TENCompetence and in other contexts, it was decided to add this to the Wookie API. The LAMS team has since been able to implement a plugin for their learning design system.

In the earlier part of the project the team had developed an internal specification for collaboration-focused widgets using shared states. This was later replaced with an implementation of the Google Wave Gadget API.

As well as the Gadgets API, integration was implemented between Wookie and Apache Shindig, which serves OpenSocial Applications. This enabled OpenSocial applications to be shared by Wookie in the same manner as W3C Widgets, extending the range of widgets available.
In addition to core functionality there were significant improvements to the user interface for the server, for example in the administration interface, with the use of AJAX and JQuery to improve the responsiveness and usability of the application.

### 4.3.3 Plugins

The principal integration developed in the project was with the Recourse learning design editor and Astro learning design runtime systems. This was largely achieved in D6.2, and only fine tuning has been required on this aspect in the period covered by this report. However a in order to demonstrate the interoperability of the widget services approach, and in order to build a user base, a plugin was developed for a leading Virtual Learning Environment (Moodle). A plugin was also developed for a leading Social Software platform (Elgg), but this was created by the Elgg community.

**Moodle**

As a useful way of testing the Widget server we developed a plugin for Moodle, which allows Widgets to be included in Moodle courses as part of the “Blocks” extension system built into the VLE.
Figure 23: Widgets in a Moodle course using the Wookie Plugin

The instructor can add Widgets by turning editing on in the usual Moodle fashion, and then choosing “Widget” from the Blocks menu. This displays a gallery of Widgets currently running in Wookie (see below) that they can choose to add.

The plugin maps Moodle roles to “Moderator” properties within Wookie, and enables functionality such as locking and unlocking widgets for users with staff rights.

Another innovation implemented in the Moodle plugin was an “embed” feature that enables Widgets – including their contextual information such as the current user information – to be taken and embedded in another environment, such as the user’s personal Netvibes page.
Elgg

Elgg is a social networking platform, and a plugin was developed by the Elgg community to show the use of Wookie widgets in a non-formal learning setting. It has since been taken over by Bernhard Hoisl as part of the Language Technologies for Lifelong Learning (LTFLL) project, has been significantly improved and extended, and is now hosted on the Elgg Community website.

4.3.4 Widgets

The team developed the Widgets described below. Some of these were developed in response to requests from designers, including the forum and chat widgets delivered with D6.2. New widget services in this release include Natter (an improved chat widget), Google Maps, the YouDecide vote widget, and WookieWiki. Others were developed with the intention of exploring or showcasing the platform rather than for any pedagogic purposes.

Natter

Natter is a chat widget developed by TENCompetence that works in a similar manner to Skype, AIM or MSN, and provides typical functionality such as graphical smilies and user avatar images. It also supports “moderator” controls to lock and clear the chat, which can be set by the plugin setting a Moderator preference value when the Widget is instantiated; in an LD system such as Recourse this is the active user role within the learning design; in other systems such as Moodle this is the static user.

Wookie plugin for Elgg http://community.elgg.org/pg/plugins/hoisl/read/323321/wookie-widgets-21
role (e.g. teacher). Note that Natter, like all the other widgets mentioned below, does not require any special server-side code to run – it only uses JavaScript and calls to the Wookie API.

![Figure 25: The Natter widget](image)

**Maps**

This is a Google Maps widget and was adapted from one developed by the PALETTE project, principally to identify any interoperability issues between the PALETTE and Wookie/TENCompetence implementations of the W3C Widgets specification.
YouDecide

YouDecide is a poll widget, where the moderator can set a question and some options, and then users can vote for their answer. Unlike most poll applications, because Wookie sends updates to widgets in real time, the results of the poll update live for each user – so the overall tally can be observed without refreshing the page. Moderators also have the option of clearing and resetting the poll, or to lock the poll to stop any more votes being cast.
WookieWiki

This is a complete Wiki application implemented as a Widget; users can edit the text using Wiki markup.

![WookieWiki](image)

Figure 29: The WookieWiki widget

Weather

Weather is a simple widget that collects weather information from the BBC’s online weather service using AJAX; the principle purpose of this widget is to show developers how to access external web services from within a widget.

![Weather](image)

Figure 30: The Weather widget

WaveWord

WaveWord is a simple turn-based word game implemented using Wookie. This was developed to explore how turn-taking can be implemented in a distributed environment without any central server managing the interactions.

![WaveWord](image)
Sudoku

Sudoku was adapted from the Google Wave Gadget used to demonstrate the Google Wave platform. Converting this to a W3C Widget and deploying in Wookie was very simple. The game is multi-player and keeps track of individual scores.

Poetry

The Poetry widget allows users to move words around to create poetry – again, this is collaborative, and the words move around in real time for each user. This creates a very dynamic feel to the collaboration as users wrestle over control of the poem. As with Sudoku, this was adapted from an existing Gadget used in the Google Wave demonstration platform.
Bubbles

Bubbles is a simple colour matching game originally developed for the Opera Widgets platform [OPERA]. It was converted to run in Wookie to identify interoperability issues with Opera’s implementation.

Further information

The work was presented at the Mashups & personal learning environments (MUPPLE) workshop at EC-TEL 2009.

A white paper produced was also produced in collaboration with the PALETTE project and was submitted to the W3C Widgets Working Group meeting in Paris in 2009.

5 Integration

Integration in WP6 takes place at two levels.
1. Between the components of the TENCompetence Learning Design Toolkit suite of applications, principally integration of the QTI assessment system with Learning Design applications.
   - Authoring: integrated authoring of services and assessment; links to repositories; the launch of UoLs in a browser.
   - Runtime: integrated delivery of QTI assessment and of the results of assessments in guiding progress through UoLs; integrated delivery of flexible services delivered through collaborative widgets; integration of LD runtime into the TENCompetence Personal Competence Manager.
2. Between parts of the TENCompetence Learning Design Toolkit and the Personal Competence Manager, which is implemented in the LifeRay portal environment. UoLs can be aligned with competences using the Personal Competence Manager. Tools are under development for storing and publishing UoLs within the Personal Competence Manager, and for populating them with groups of learners. UoLs can also be published and populated with learners directly from the ReCourse editor.
In the period covered by this report work has been carried out on both aspects, but with a greater emphasis on the first aspect.

5.1 Integration of the components of the Learning Design Toolkit

5.1.1 Integrated authoring

While the ReCourse Learning Design Editor appears to the user to be a unitary application, under the surface it leverages the Eclipse Java plug-in framework to integrate and the servers of the TENCompetence Learning Design Toolkit to provide integrate a range of components. In D6.2 we described how ReCourse supported authoring of remotely hosted Wookie widget services, how a QTI authoring plug-in was provided, and how provisioning of the LD runtime system could be carried out from within the authoring tool. This integration is extended in the present system.
As regards authoring, it was observed while access was provided to integrated functionality, authors were confused by the types of elements which they could include as learning objects within a Unit of Learning. For example, a QTI test was shown simply a resource, and superficially appeared the same as a web page. To articulate the distinctions between learning resources, executable tests, and widget delivered services the following new Learning Object types were added to the Environment Editor in ReCourse: knowledge object, test and tool.
This was not simply a matter of adding a plug-in to the code base. The objects were then linked to the relevant functionality, so that a Knowledge Object enables the author to select a resource in the file library of ReCourse (or add a link), and a tool is linked to the remote Wookie Widget service. However, the most significant work in the present release of the authoring software has been in linking a test object to a QTI test, to be run by APIS. This is complemented by enabling tests to be created in the activities editor “New Test Activity”, and in the Resources tab “New QTI Test Resource”. We now describe this functionality, and it should be remembered that the process described involves two separate applications integrated through the Eclipse plug-in structure.
Add a QTI Test as a Resource.

From the Resources section in ReCourse add a menu item to create a new QTI Test as a Resource in the UOL:

![Figure 34: Adding a QTI test as a resource](image)

This does the following:
1. Creates a new, blank QTI Test XML file in the UOL’s main folder
2. Creates a new Resource type in the UOL that references the newly created file
3. Opens the new QTI Test document in the QTI Editor.

Add a "New Test" type Activity and wizard

From the Activities view in the “Library” and from the “Design” tab add a menu item and wizard to create a new “QTI Test” type Activity to the UOL:

![Figure 35: Create new test activity](image) ![Figure 36: New Test Activity wizard](image)

This does the following:
1. Invokes a wizard to create a new QTI Test type Activity that references an existing QTI Test Resource (added previously) or to create a new QTI Test file. A Role may also be selected to create
Role Parts if adding in the “Design” tab.

2. If not referencing an existing QTI Test file:
   a) Creates a new, blank QTI Test XML file in the UOL’s main folder
   b) Creates a new Resource type in the UOL that references the newly created file
   c) Then do the following stages in item 3.

3. If referencing an existing QTI Test file:
   a) Creates a new Learning Activity.
   b) Creates an “Item” type that references the Resource and adds this to the “Description” element of the Learning Activity
   c) Opens the QTI Test file to get its internal identifier
   d) Creates a new “Local Personal” Level B Property based on the identifier and adds it to the Properties model in the UOL.

4. Opens the QTI Test document in the QTI Editor.

Add a “New Test” type Learning Object in Environment Editor

From the “Environments” tab add a palette item to create three distinct types of Learning Object – “Knowledge Object”, “Tool Object” and “Test Object”:

![Figure 37: adding a new Test Object to an environment](image)

Creating a new “Test Object” type does the following:
1. Adds a new Learning Object type to the Environment in the UOL and sets its type to “Test Object”
2. Creates a new, blank QTI Test XML file in the UOL’s main folder
3. Creates a new Resource type in the UOL that references the newly created file
4. Creates an “Item” type that references the Resource and adds this to the “Resources” element of the Learning Object
5. Opens the QTI Test file to get its internal identifier
6. Creates a new “Local Personal” Level B Property based on the identifier and adds it to the Properties model in the UOL.

**Recognise QTI types for display and editing and other internals**

Many QTI Utility functions have been added to the ReCourse code base in order to achieve this integration. This includes methods to recognise QTI file and resource types, and extensions so that the QTI editor can integrate menu items and file handlers.

**5.1.2 Integrated of QTI authoring and runtime**

NewAPIS now renders all the types of questions which can be created by the QTI editor, and input and output problems between the two applications have been solved. A particular problem has been handling incompatible codification of accents, an issue which has now been resolved.

**Report results: nselected and npresented variables**

The variables (nselected and npresented) are now included in each test since they are necessary for newAPIS to know when a test is over.

**5.2 Integrating ReCourse with Fedora (Task Force 13)**

In line with the requirements of Task Force 13 - ReCourse linking to Learning Objectives/Prerequisites and "Search Link to a UOL" functionality has been built into the ReCourse Editor which enables Units of Learning (UOLs) to be packaged into zip files and uploaded directly to the TENCompetence Fedora database from within ReCourse.

The focus of Task Force 13 has changed since its formulation, as the Learning Path Editor no longer searches the Fedora database of LearnWeb2.0 for appropriate Units of Learning, but rather the database of published UOLs on the Personal Competence Manager (PCM) CopperCore server database. Using the Learning Design portlets for the PCM which are currently under development, users with sufficient privileges will be able to search Fedora from within the PCM, identify UOLs which they want to use, and publish these on the PCM. These published units of learning are then related to competence development goals by means of the learning path editor tool in the Personal Competence Manager. Thus the development of the Learning Design Administration Portlet and link tool forms an important part of the revised approach to defining the competence development goals of UOLs.
5.3 **Design of portlets for the integration of the LD Toolkit into the TENCompetence LifeRay based Personal Competence Manager.**

Between parts of the TENCompetence Learning Design Toolkit and the Personal Competence Manager, which is implemented in the LifeRay portal environment. UoLs can be aligned with competences using the Personal Competence Manager. Tools are under development for storing and publishing UoLs within the Personal Competence Manager, and for populating them with groups of learners. UoLs can also be published and populated with learners directly from the ReCourse editor.
6 Brief closing remarks

In this deliverable we have reported on substantial development work, leading to major new releases of key Learning Design Toolkit software, in particular ReCourse Learning Design Editor v.2.0 with its QTI plug-in, the final version of the Wookie Widget Server, and the new APIS runtime system.

In the final months of the project these systems will be refined and documented, and the main focus of development work will shift to finalisation of the Astro Learning Design player, and integration with the Personal Competence Manager.

The Wookie Widget Server has attracted interest from the Apache Foundation, and it is planned to complete documentation of Wookie and submit it to the Apache Incubator before the end of the project.