2008

Using widgets to provide portable services for IMS Learning Design.

Paul Sharples  
*University of Bolton, P.Sharples@bolton.ac.uk*

David Griffiths  
*University of Bolton, dai.griffiths.1@gmail.com*

Scott Wilson  
*University of Bolton, S.Wilson@bolton.ac.uk*

Digital Commons Citation  
http://digitalcommons.bolton.ac.uk/iec_conference/5
Using Widgets to Provide Portable Services for IMS Learning Design

Paul Sharples, David Griffiths, Scott Wilson
The Institute for Educational Cybernetics, The University of Bolton
Bolton, United Kingdom BL3 5AB

P.Sharples@bolton.ac.uk
D.E.Griffiths@bolton.ac.uk
S.Wilson@bolton.ac.uk

Abstract
Since the publication of the IMS LD specification it has been recognised that the lack of a rich set of runtime services is a major barrier to adoption. The approaches taken to resolving this problem are reviewed, and their strengths and limitations identified. A generic widget server developed by the authors is described. Integration of the widget server with the IMS LD runtime system provides an extensible set of services. This has been demonstrated with the creation of widgets for forums, messaging, vote, and Google maps which are provided for users within the context of their role and activities in a Unit of Learning. Authoring and administration are described, showing how the system is both extensible and portable.

Keywords
IMS Learning Design, interoperability, integration, widget, portable services

1 The problem addressed

IMS Learning Design (IMS-LD) is a specification developed by IMS Global Learning Inc [1] which enables users to implement learning activities for multiple users while maintaining the flexibility to define a wide range of pedagogical structures, and run them on any compliant application. A comprehensive introduction to the specification is available in Koper [2]. As discussed in Griffiths and Liber [3], IMS-LD has in some respects not achieved the levels of adoption which were initially anticipated. One reason for this is that the ability of IMS-LD players to orchestrate runtime services has so far been extremely limited. This problem was recognised when the specification was developed, as recognised by Olivier, one of the authors of the specification [4], p.38:

Clearly many more services could be added to the LD specification, and it is desirable that they should be, from chat, instant messaging and white boards, through virtual classrooms and more sophisticated collaborative services, such as virtual design environments, to sophisticated simulation and multi-user game-playing systems.

The key issue that needs to be addressed is how to add services in such a way that key learning designs that use them still retain a reasonable degree of portability across different LD-compliant platforms. If all the above services were included, could any system be expected to be compliant? Or should the specification stick to the lowest common denominator for services...

IMS-LD does not define the actual services for use within a learning context, such as a wiki or forum. Rather, it specifies a small subset of four generic service types that could be used within a learning context:

- Conference
- Monitor
- Send Mail
- Index Search

Consequently the question arises of how the design is to be realised in runtime software. Most interestingly, what does a conference consist of? Is it voice chat, instant messaging, video conferencing or something else? What happens if I want to use service X in my Learning Design?
2 Some existing approaches to implementing services in IMS LD

Olivier and Tattersall also comment that “Learning services are likely to come in two varieties: those ... which are set up as part of a local environment; and those that are set up as remote web services” [4] p.39. These varieties are also termed tightly and loosely coupled scenarios, and systems to date have implemented the latter approach. This was analysed by Tattersall et.al. [5] in relation to the integration of a SCORM player with IMS LD runtime, describing the use of a Dispatcher to direct resources to the appropriate runtime system. A system was implemented following this approach by the present authors, described by Sharples [6] A similar approach was used by the developers of the SLeD LD Player to use Moodle as a provider of services, as announced in [7] using the approach described in [8], and demonstrated with an integration of the Moodle forum.

This approach provided effective integration for both SCORM and the Moodle forum, but did not provide a generalisable solution. Firstly, the effort involved in carrying out the integration was substantial, and the work needed to be repeated with each individual service to be integrated. Indeed Little [9], responsible for the Moodle forum integration, comments that many cases “you would probably want to link up sled to your own actual service providers (esp. for the forum)” and that this “would not be insignificant work”. Secondly, it was not portable, indeed when a new version of Moodle was released the integration no longer worked, creating maintenance problems which appear to have been insurmountable.

Another approach is to build the player within an environment which makes a rich set of services available. This is the case for the IMS-LD player in the .LRN Virtual Learning Environment (VLE) [10]. In this case the player can directly map the IMS LD asynchronous conference service to the forum service available in the .LRN environment, and a similar solution is available for sendmail. This solution is effective, but not portable. It also ties the user to the set of services available within a particular VLE, as recognised by Escobedo et.al. [10] who describe how the inclusion of a chat service depends on its availability as a .LRN service.

Other systems have focused on tools for integration and management of services, and left possible integration with IMS LD for a later stage. This is the case of LAMS [11], which was ‘inspired’ by IMS LD, and which is able to export to IMS LD level A. However, while it provides a rich set of services and its own tools API, the information relating to the use of these tools in UOLs is stripped out on export to IMS LD, and so it does not provide a solution to the problem under discussion.

3 The solution developed

While these approaches to implementing services are satisfactory for their own purposes, none of them provide a solution to the problem identified by Olivier, i.e. the provision of a rich set of services in IMS LD while maintaining portability across different LD-compliant platforms. The solution proposed here was developed within the TENCompetence project, and centres on the development of an architecture which supports collaborative widgets, integrated into the existing IMS-LD infrastructure. This loosely coupled system, is both portable and extensible. We have developed a server implementing this approach, code-named Wookie, and have reported on its architecture [12] and implementation [13]. It should be noted, however, that although the widget server was developed with the needs of IMS-LD in mind, it has been created as an independent server, and integration has already been demonstrated in Elgg and Moodle. Here we focus on the use of this server to provide extensible and portable runtime services for IMS-LD

4 Integration of the Widget Server with IMS-LD runtime

The Widget Server is responsible for providing a particular widget requested by the IMS Learning Design runtime system. Consequently the Widget Server needs to know, which specific widget to supply back to the running UOL. A solution to this was formulated whereby existing elements and attributes of the IMS Learning XML binding could
be used to identify these specific widget services. For each IMS-LD service a parameter value can be specified. This may be any text an author wishes, but is usually a name-value-pair. To use one of the widgets made available from the widget server, a parameter must be added to an existing service element within an environment in a UOL. The name-value pair string to enter takes the following syntax, widget=<type of widget>. So for example, to use the default chat widget service, one would enter widget=chat. Similarly to use the default forum widget, one would enter widget=forum.

We focused on the use of the “Conference” service, since the “Monitor” and “Index-Search” are IMS-LD specific in their use. “Send mail” is associated with an email service, but in theory this too could be realised by use of an email widget.

The realisation of the widget system working along side CopperCore and SLeD, meant that some other changes were needed in those systems. Firstly, a new service was needed within CCSI. This service is responsible for propagating the requests for a widget from the SLeD player, to the Widget Server. Additionally the SLeD player also needed to be updated, so that it could handle the widget=contex parameter and hand it off to the new CCSI service.

Figure 1. Integration of CCSI and Widget Service

Finally, SLeD had to be able to parse the reply from the CCSI service and show each widget it found, in a draggable, resizable window. The solution chosen for this was to use Fenster which is a cross browser Javascript windowing library.

5 The interaction cycle

The SLeD player provides the user interface for the Unit of Learning. When a user logs in they navigate around the UOL until a service is encountered within a particular environment. When SLeD obtains the environment information from the CopperCore engine, it first parses each service entry found, to see if it contains any widget=context entries. If one is found, then the SLeD player builds a query. The query contains information which is specific to the run, environment, service and user who is requesting the widget. Next, SLeD passes the query to the CCSI widget service, which in turn calls the Widget Server. The Widget Server then uses these parameters to either return an existing widget instance (an instance which has been used before) or create a new widget instance. The CCSI widget module takes an xml response returned by the Widget Server and passes those values back to SLeD. The SLeD player can now translate this information into the user interface. For example, returned values contain the URL where the widget can be found and the widgets height and width to be displayed. SLeD now creates a Fenster window instance in the browser. Ultimately, once the user clicks on the widget link in the browser, a pop up window appears containing the widget content.

6 The extensibility of the Widget services

The Widget Server has an administrative section where new widgets can be added by importing an archive package containing all of the widget’s resources. The archive also contains a manifest called config.xml, which must conform to the W3C widget manifest format. Once successfully imported, the administrative user can assign and create widget contexts to the widget. The server can host a number of different widgets for the same service, and in order to support portability the administrator designates one of these as the default for a category of widget. For example, if a UOL specifies a particular chat widget which is not available on the runtime server, then the learner is provided with the default
chat widget. The integrated TENCompetence IMS-LD runtime system ships with a number of default widgets, including chat, forum and vote.

Consideration was given to how widgets would be authored at design time. It was apparent that authors would have difficulty in remembering to enter the `widget=context` parameter manually at the design stage. Additionally, authors had no way of knowing which widgets were available to use from a given Widget Server. To overcome this, an advertising service was written into the Widget Server. An authoring tool can query the widget advert and provide the author with a list of available services. This link between Widget Service and authoring tool has been implemented within the TENCompetence IMS Learning Design authoring tool, ReCourse.

ReCourse utilises the widget advert within its `environments` editor. An item in the tools menu, can query a Widget Server, to see which widgets it can provide. ReCourse uses the results of the advert to show a graphical icon of the widget, which it places on the tools palette. The icon location is also part of the advert, as each widget has an associated icon. When an author wishes to use a widget, they simply drag and drop the widget icon from the palette onto the canvas. This creates the appropriate service within the Learning Design, with the `widget=context` parameter automatically set.

7 Conclusion

The system developed provides a solution to the problem set out by Olivier. It constitutes a framework for the implementation of an extensible set of services for IMS-LD, a means of managing and authoring them, and a basic set of default widgets. The system is currently being piloted within TENCompetence. The availability of this new functionality has implications for the way in which teachers and learners work with UOLs which we will examine in future publications.

Acknowledgement

The work on this chapter has been sponsored by the TENCompetence Integrated Project that is funded by the European Commission's 6th Framework Programme, priority IST/Technology Enhanced Learning. Contract 027087, www.tencompetence.org.


Volume,


