Scoping the future: A model for integrating learning environments

Niall S F Barr and Sarah L Honeychurch
University of Glasgow, Scotland

Abstract
The Virtual Learning Environment (VLE) has become synonymous with online learning in HE. However, with the rise of Web 2.0 technologies, social networking tools and cloud computing the architecture of the current VLEs is increasingly anachronistic. This paper suggests an alternative to the traditional VLE: one which allows for flexibility and adaptation to the needs of individual teachers, while remaining resilient and providing students with a seamless experience. We present a prototype of our vision, combining our new development software and a number of existing tried and tested tools into a single flexible interface, and built on established pedagogical and technical standards.

The origins of the modern VLE
The modern concept of the Virtual Learning Environment (VLE) or Learning Management System (LMS) first appeared soon after the advent of the World Wide Web, which made the Internet accessible to non-technical users. Prior to that online learning systems had required custom software on the user's computer, or considerable technical skills from both the students and their teachers, and so were much more specialised than the modern VLE. A VLE is simply an interactive website where software running on the server creates web pages on-the-fly in response to a client request. One of the first systems to make use of web technology for teaching was Clyde Virtual University (CVU) (Whittington and Sclater, 1998) developed by Strathclyde University. CVU contained web pages for individual courses which were only available to registered students, and which contained links to pages of learning material, quizzes and discussion forums.

More than 15 years later, surprisingly little has changed in the VLE. The typical format still consist of course pages which list relevant information, and links to web pages, forums and quizzes. Newer additions to this list include assignment upload, wikis and support for peer reviewing. Although each of these functions is quite separate from the others, all of them were built into the same large software system, typically either running on a single server or with servers split between database and other functions. These VLE systems typically take on the full online course material delivery role at the University, however they are quite separate from back-end systems that deal with student course selection and grades.

Current UK situation
In the U.K. higher education sector two VLE platforms, Moodle and Blackboard, each have a substantial share of the market, while a number of other platforms including Sakai, Desire2learn and local customizations of Sharepoint are used in very small numbers of institutions. Moodle and Blackboard both provide similar facilities for teaching staff and students. For support staff the platforms are very different, as Moodle is open-source, and adding custom features is fairly simple. Blackboard is proprietary software, and an additional licence is required to carry out any customization.

Issues with current VLE provision
While these VLEs have been very successful, and are widely used almost all universities, they do have a number of limitations. It is difficult to customise the VLE to suit specific needs of different courses, as a single instance is generally used across the entire institution, and in addition the individual components of the VLE, such as wikis or quiz systems, inevitably trail behind the features of the best specialised systems for these tasks. Where the VLE is locally hosted, a further problem faced by computing services is the issue of scalability and

predicting future use. The VLE is a major application which needs to run on high specification hardware, which will be expected to last for several years. The level of use of VLEs has increased dramatically over a fairly short period, and so making predictions about the future hardware requirements is not easy. An alternative approach is to run a large number of small servers, connected to appear as an integrated whole. This approach requires software designed to run as separate small services that can be distributed over variable numbers of servers. Recently this approach has been popularised as cloud computing, however the concept of a VLE running as a large number of small services has been around for some time.

A previous attempt to challenge this: JISC eLF

The JISC e-Learning Framework, or eLF, (JISC 2004) was intended to address the need for methods to integrate systems supporting both institutional processes and e-learning, and provide a structure that would be flexible both for pedagogy and technical innovation. The vision was for a large number of service applications, each addressing a relatively small, specific need, and communicating through SOAP web-services. This would mean, for example, that a student taking an online quiz might login using an 'Authentication' service, be guided to the assessment through 'Sequencing', and 'Activity Management' services, and then take the quiz using an 'Assessment' service. A number of projects were run, which created some of these services (and in the case of Assessment, at least three similar variants of the service) however a coherent whole failed to materialise. Fundamentally, although a good concept, the vision of eLF was too complex and the projects lacked well defined standards for communication between modules.

One of the key features of the JISC eLF was to be the use of standards where applicable, for example assessment systems were designed to be compliant with a draft version of IMS QTI 2.0. However, an appropriate standard for communication between modules was lacking. The use of SOAP web services to provide this communication was (at the time) an obvious choice, however without considerable detail about the data structures that would be passed over the SOAP communications it would be impossible to design interfaces for interoperable services in isolation.

Our Backgrounds

The presenters approach this challenge from a blend of perspectives: one is a University tutor who also works as a learning technologist; the other is a professional programmer with 25 years of experience of teaching in HE. We thus bring together pedagogical and technological considerations and incorporate both strands into one conversation.

I (Niall) have been working with learning technology for over 20 years, and when appropriate also getting involved in teaching. Initially I worked with desktop software which supported small well-defined sections of learning, sometimes written by myself and sometimes from third parties. Some of these pieces of software were intended to be complete tutorials on themselves, while others were simulations that could be used as the basis for laboratory style tutorials. While some of these pieces of software were very useful, it was clear that there was a very high ratio of software production cost to the level of use the software would get. It was certainly not viable for a single institution to be developing this type of teaching material for just their own students. The advent of the World Wide Web showed a way of producing courseware more rapidly, whilst also making it available to a much greater number of students. In 1996 I took the lead on a project developing web tutorials for anatomy, which were to be used by students at all universities in the West of Scotland. Our package included web pages containing text, illustrations, animations and videos to teach students about functional anatomy and biomechanics, and interactive quizzes for self-assessment, however other VLE features were completely missing. Frustratingly, no real assessment was done with regard to the effectiveness of this teaching material, although it continues to be available for students. My next job was as a teaching assistant for a

postgraduate IT course, where I spent my time in the computing lab providing one-to-one help as required. I realised that in this context the experience of the tutor is essential, because there are a huge number of ways in which a student can be stuck with a problem and without the guidance of the experienced tutor a relatively minor issue can become a major blockage. This highlighted to me how much difference there can be in the requirements of different subjects. Previously, teaching aspects of physiology or biomechanics to zoology students I had to explain difficult concepts, however I never saw zoology students being blocked by a single small problem.

More recently I have switched from being a developer at a department level to being part of the central support team for Moodle at the University of Glasgow, where I am mainly involved in developing new features. Moodle is a large piece of software, with a huge amount of code loaded into the server's memory every time a page is viewed. Developing new components for Moodle requires not only following the coding standards, but also making use of the underlying code wherever possible. Although the Moodle coding standards are largely sensible guidelines which prevent security issues, having to adhere to coding standards and make use of extensive existing libraries like this also make it harder to integrate software from other sources. The design of Moodle means that we cannot reasonably provide different facilities to different teaching units, and more importantly that we must be very certain of the reliability of any component that gets added to the system.

My experience providing support both at a departmental level and as a member of central services has led me to believe that we need a more flexible system, where the core features supported at university level can easily be mixed with features supported at a department level (but posing no risk to the centrally supported features.) I have also come to believe that we need to be able to embrace new ideas more rapidly, by making use of smaller specialised pieces of software rather than large relatively difficult to maintain software where appropriate.

I (Sarah) began as a Graduate Teaching Assistant (GTA) in Philosophy at Glasgow over ten years ago. We Philosophy GTAs have a fair amount of autonomy with regard to how we deliver our teaching, but the primary method is weekly face-to-face tutorials of groups of about 15 students. Philosophy has a Moodle course for each of its pre-honours courses, but these are mainly used to host lecture slides and other documents. For the last few years, the course convenor has also provided each GTA with a tutorial group forum and encouraged us to use these to inform our students about tutorial timetables, topics and office hours. Occasionally students will post a message to one of the forums, but on the whole students are passive consumers rather than active participants.

Three years ago, for a variety of reasons, I began to want a more structured approach to my tutorials, and in July 2010 I secured funding from the (sadly now defunct) Higher Education Academy Subject Centre for Philosophical and Religious Studies in order to develop collaborative learning techniques for my tutorials (Honeychurch, 2012). I call the method I developed for this project Jigsaw wikis (this method was inspired by Aronson's jigsaw classroom technique, see Aronson 1978 for example). My initial vision was one where I would deliver tutorial questions to groups of eager students prior to the tutorial, and small groups of students would somehow collaborate online in order to fill in the answers to all of the questions prior to arriving at the face-to-face weekly tutorials. This turned out to be over-ambitious, although some aspects of the technique were successful. I have since had time to reflect on my project and think about why it might not have worked as I initially expected.

When I devised my project in 2010, I had very little experience of using Moodle (I only had non-editing staff rights to Moodle courses) and no background in learning technology. This led to me making assumptions about how easy Moodle was going to be to use, both for me

and for my students. In particular, I assumed that students would be far more competent at using Moodle than they were, and also that Moodle was far more flexible than it actually is.

I assumed that because my students had grown up with web 2.0 technologies such as Wikipedia and Facebook, they would all be “digital natives” (Prensky, 2001): that they would find Moodle easy to use and would be confident in writing forum posts and wiki pages. In fact, very few students posted to the forum or edited the wiki pages, although they did use the wiki in order to download questions into a word processor and write their own, private, answers. Feedback given to me at the end of the course taught me that many students are not as digitally competent as I thought, and that they see a big difference between using the technologies they are familiar with (such as Facebook) and others such as Moodle. Although I firmly believed that the same skill set was needed, I had no way of tapping into their familiar media and embedding it into my own Moodle course.

In late 2010 I began working at the University of Glasgow as a learning technologist, and I now support staff and students on a day-to-day basis (part of my job involves answering staff and student queries sent to our help desk). This has taught me that my students and I were not unusual in our attitudes towards the VLE. Moodle 1.9 has a dated look and feel, with few of the web 2.0 features that modern students and academics have come to expect. Moodle 2 is little better. It suffices as a place for staff to upload resources and students to submit assignments, there is limited potential for self and peer-assessment, but there is little incentive for most academics to expend effort in building and maintaining courses there.

Requirements

Recently the concept of cloud computing has become popular, where large numbers of relatively low powered physical or virtual servers are used together to provide a flexible and scalable solution. Although cloud computing is normally associated with hosted services such as Google docs, or hosted virtual computing and storage such as Amazon EC two or Microsoft Azure, the same approach can be used to provide flexible scalable computing facilities locally.

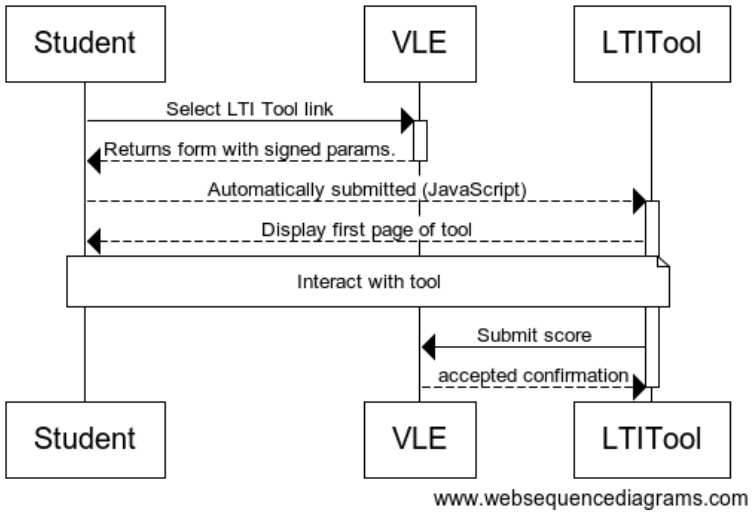
By making use of some of the ideas of cloud computing it should be possible to create a more flexible system than the current generation of VLEs. As the VLE has become a core part of the systems at university level, it has become necessary to treat changes to the VLE with the same caution as with core business systems. In effect this means a VLE which cannot be customised for specific needs, and is only upgraded at widely spaced intervals. This also means that when it is necessary to change provider or make a major upgrade (for example from Moodle 1.9 to Moodle 2), it is difficult to stage a gradual migration. These thoughts have led us to come up with a list of requirements for a future VLE capable of replacing the current generation with a more flexible alternative.

- The system should be designed to run either on a single large server or on multiple small servers or on a mixture of the two.
- It should be easy to extend the system as required by adding further servers.
- The system should support gradual revolutionary change change, with the possibility of mixing more than one version in in a university's system.
- It should be possible to add external tools, so that an individual course might consist of core tools hosted within the central IT systems, local specialist tools hosted in the Department's own server, and external specialist tools hosted with publishers or on commercial hosting services. The link to these external tools must be designed so that there is no risk of the external tools interfering with core services.
- All personal/academic data must be held in safe locations, with the minimum necessary being passed to external tools.
- The system needs to be flexible in order to:
    - support the best emerging practice,

- adapt to the changing needs of the academic community,
- accommodate the diverse pedagogical requirements of a multi-disciplinary institution.

Standards that can underpin a new approach

A key requirement for systems which are to be used over a long period, or which will host material which will be needed over a time period greater than the lifespan of the system is that they have good interoperability. Whilst most of us rarely consider the issue of software interoperability, we are very dependent on it. We routinely exchange documents in Microsoft Word format, which has become the de facto interoperability format for word processing, and colleagues making use of alternative word processors are able to open and edit the same documents. All digital cameras support the same variant of the JPEG file format, meaning that our digital photographs are easy to share and open with any standard photo viewing or editing software. Interoperability standards specific to education are less well-established however a number do exist. Of particular interest to our vision of a more flexible distributed VLE is the IMS Global Learning Consortium's Learning Tools Interoperability (LTI) specification. (McFall et al. 2012) LTI is a lightweight single sign-on system based on the well-established OAuth specification which allows external websites to be securely launched from a VLE with the minimum transfer of data needed.



LTI Launch Sequence

We have already implemented a slightly modified version of LTI to allow courses in multiple instances of Moodle to be in a customised "My Courses" list on each Moodle instance, with single sign-on enabling students to transfer between Moodle servers without having to login again. This is facilitating a staged migration from Moodle 1.9 to Moodle 2.3, without requiring students to be fully aware of the separate servers. A further enhancement intended to go live this summer replaces the navigation through categories to find courses in Moodle with a separate system that links directly into courses in in several different Moodle servers. As well as assisting with the staged migration process, this tool is expected to reduce overall server load as it is able to make use of a much simpler authorisation system than Moodle (which has to manage multiple different roles with different levels of editing and viewing rights.) Whilst developing this new front-end, we realised that there was no fundamental reason why it should be used only with Moodle, and other systems supporting the same modified version of LTI could also act as individual course servers as part of a near seamless integration.

While LTI has provided us with the basis for communication between modules a modular VLE, further interoperability standards are also incorporated in our vision. The IMS Common Cartridge (Kahn 2011) specification provides a basic format for transferring the learning material for modules and courses between VLEs. Common Cartridge includes links to external LTI resources, and so is very compatible with our vision. Common cartridge also

includes a profile of IMS QTI version 1.2 for exchanging simple assessments, however we believe that QTI 2.1 (Kraan et al. 2012) is now ready to become the standard for this type of assessment. The e-assessment projects that were part of the JISC eLF provided the basis for a number of further projects, and two of the most recent projects, QTI-DI and Uniqurate have delivered the basis for a flexible open source LTI connected e-assessment platform which is substantially more powerful than existing systems.

Our university, like many others, has back-end systems that support the IMS Enterprise Web Services and Learner Information Profile (LIP) specifications. These will provide the basis for communication between the modular VLE and backend management systems, acquiring information such as student course enrolments, and returning grades. Although LTI provides the basis for transferring logins between modules, some further information also will need to be transferred to make a fully integrated system. We have started defining some simple REST web services to handle this information.

Other emerging standards may also be of great relevance, in particular ePub version 3 which looks likely to become a popular standard for delivering textbooks to multiple different types of digital reading system. EPub 3 is built on an XHTML representation of HTML 5, and so is very compatible with a web delivered platform.

Our structure

The structure of our modular VLE consists of a front-end application which communicates with the central information management system to acquire enrolment information, and which broadly resembles the course structure areas of Moodle. Links from this application use LTI (with optional extensions) to provide a single sign-on into separate course delivery applications. While our system has its own specific lightweight course delivery application, Moodle could also be used. The course delivery application is able to directly provide a basic overview of the course and also static web page content. All more interactive features such as forums, wikis and quizzes are delivered through separate LTI applications, again with optional extensions to a greater appearance of integration. In the current prototype the course page is a fairly static page, not unlike that in Moodle, however we are also considering an option of a main course page which is modelled more on a Facebook group page. As well as using LTI to link to external modules, we are exploring the use of widgets to provide more personal tools such as bookmarking, personalised dictionaries, or optional facilities such as feeds from discussion forums.

Custom extra parameters for LTI

The main original use case of LTI was to provide a way for publisher hosted content to be securely accessed from university VLEs with the minimum necessary information being passed to the external publisher. The standard LTI parameters include information about the context of the launch, a context specific user identifier, and information about the user roles (such as learner or instructor). LTI makes no guarantee that the user identifier identifies the same user in different contexts, and more user specific information (such as name and e-mail address) are optional parameters that can be passed to more trusted tools.

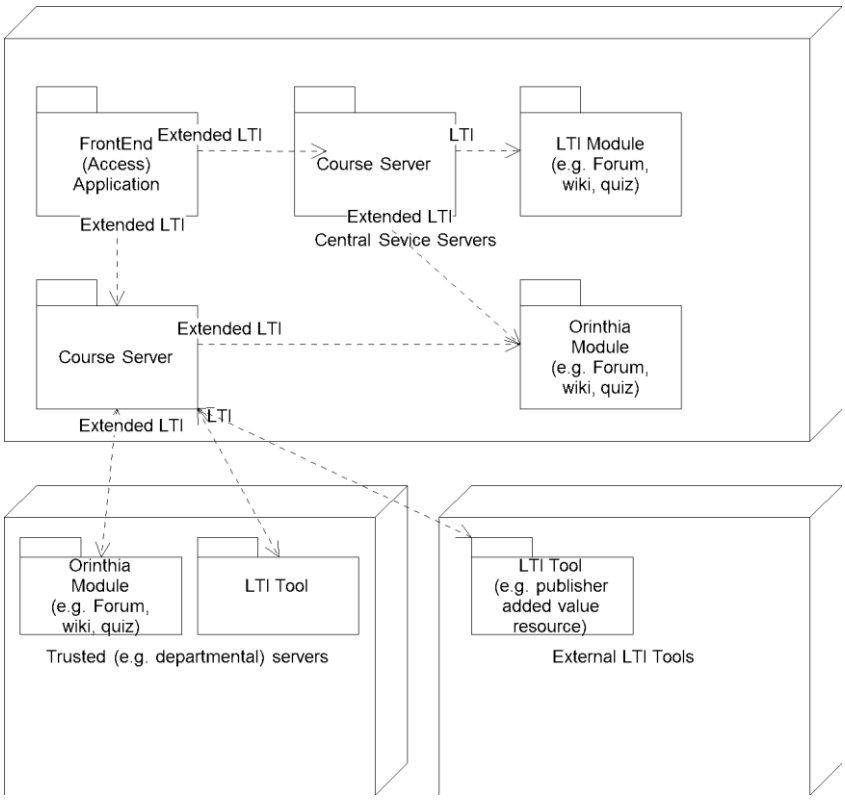
In our distributed VLE vision we will sometimes require more detailed information about the user, and would also like to support other features such as common page templates and stylesheets to provide a coherent user interface, and breadcrumb trails to provide navigation. For this reason we are in the process of defining a set of extension parameters to LTI which will allow a tool to appear to be a more integrated part of the system. However, it is important to note that these will be optional parameters, and standard LTI tools will also work with our system.

Our extension parameters will include:

- A parameter containing a URL for the tool to retrieve activity settings.

- URLs for an HTML template and CSS to allow tools to follow the main course page appearance.
- An organisational level user ID, which can be used for reporting back to a central gradebook for example.
- A breadcrumb trail, allowing the user to navigate back.
- Personal settings, which may include accessibility information and links to private bookmarks or notes.

LTI is an evolving specification, and if any of our extensions are duplicated by new features of LTI the extension will be deprecated with the standard feature replacing it.



Distributed Modular VLE Deployment Diagram

Benefits

There are a number of benefits that our modular approach to VLE provision will be able to deliver:

- The use of LTI, which is a lightweight standards based protocol, means that it is easy to add interfaces to external tools. Our design allows for the best external tools to be utilised where needed.
- The VLE becomes a collection of small extremely orthogonal modules that can be maintained individually, simplifying development and allowing greater flexibility. Modules can be written in different languages and deployed on different platforms - our prototype's course module is a LAMP (Linux/Apache/MySQL/PHP) application, while the assessment module is a Java/Tomcat application.
- Where teachers have different preferences, or subject areas have different requirements, it is possible to use completely separate modules that fit with particular needs.
- The system is extremely scalable as it is built on a collection of small servers (like Google) rather than a single large server. As capacity requirements grow extra low-cost servers can be added to share the load. Where the extra capacity requirement is temporary, or appears rapidly, it is possible to make use of virtual Cloud servers from Amazon or Microsoft Azure on a pay by the hour basis.

- Because the system is built on large numbers of small machines, and single large purchases are not required, planning can take place over shorter timescales.
- Being a highly modular system based on publicly available standards, and designed to be a collection of small open source projects rather than a single large open source project, there is very little risk of getting trapped in a vendor tie-in situation.
- The modular approach simplifies management of users' roles, which in turn reduces processing overhead. This means that our system is likely to be usable for MOOCs as well as for replacing a conventional VLE.

The future

Our prototype, which we have named Orinthia[1], is still a proof of concept piece of software. Some aspects of the code will need change before widespread use, however we believe the core design is solid. The current LTI implementation is LTI 1.1 with extensions, however LTI 2.0 is likely to slightly reduce the need for extensions so will be adopted soon. Our main components have been designed using a 'Product Family Engineering' approach, where a combination of shared libraries and generated code I used to create related applications, and we expect this will be the approach taken to create other modules designed specifically for the system. These modules that make use of our customer LTI extensions will be known as 'Orinthia modules'. They should also be able to work with any other LTI enabled system, but will have a more integrated appearance when used with Orinthia.

References

Aronson, E. (1978) *The Jigsaw Classroom* Beverly Hills: Sage http://www.jigsaw.org/

The Economist. (2010) "'Technology and society: Is it really helpful to talk about a new generation of "digital natives" who have grown up with the internet?'". http://www.economist.com/node/15582279?story_id=15582279 Accessed 05/05/13

Honeychurch, S. (2012) "Taking Forward the Jigsaw Classroom: the Development and Implementation of a Method of Collaborative Learning for First Year Philosophy Tutorials." *Discourse* 11, 2 http://www.heacademy.ac.uk/assets/documents/subjects/prs/PrsDiscourse/Discourse-11-2/Disc-11-2-honeychurch.pdf Accessed 05/05/13

JISC (2004) "e-Learning Frameworks and Tools programme" http://www.jisc.ac.uk/whatwedo/programmes/elearningframework.aspx Accessed 04/05/13

Prensky, M. (2001) "Digital Natives, Digital Immigrants." http://www.marcprensky.com/writing/prensky%20-%20digital%20natives,%20digital%20immigrants%20-%20part1.pdf Accessed 05/05/13

Kahn, J. ed., (2011). *IMS GLC Common Cartridge Profile: Overview*, Available at: http://www.imsglobal.org/cc/ccv1p2/imscc_profilev1p2-Overview.html Accessed July 2, 2012.

Kraan, W., Lay, S. & Gorissen, P. eds., (2012). IMS Question & Test Interoperability version 2.1. Available at: http://www.imsglobal.org/question/#version2.1 [Accessed April 25, 2013].

McFall, G. et al. eds., (2012). *IMS GLC Learning Tools Interoperability Implementation Guide Version 1.1*, Available at: http://www.imsglobal.org/LTI/v1p1/ltiIMGv1p1.html Accessed July 2, 2012.

Whittington, C.D. & Sclater, N., (1998). Building and testing a virtual university. *Computers & Education*, 30(1-2), pp.41–47.

[1] Orinthia is a name coined by George Bernard Shaw, who also coined the word Moodle - "intr. To dawdle aimlessly; to idle time away. Also with about, on." (OED)