Moodle Wave: Reinventing the VLE using Widget technologies

Scott Wilson , Paul Sharples, Dai Griffiths and Kris Popat,
University of Bolton, UK
{s.wilson, p.sharples,d.griffiths,k.popat}@bolton.ac.uk

Abstract. Can widget technologies revolutionize the VLE and challenge the dominant design of elearning? The authors describe an experimental implementation of the Moodle VLE using the W3C Widget specification and Google Wave Gadget API to replace existing core features. This raises four challenges; (1) By replacing most of the functionality of the VLE with simple reusable applications, the core functionality and design of the VLE is challenged; (2) by allowing these applications to be extracted and used in personal learning environments, the hegemony of the institutional system is challenged; (3) by conducting learning activities in small discrete applications, the model of student tracking is challenged; (4) synchronous, live updating applications challenges the model of user behaviour for the VLE.

Keywords: Widgets, Personal Learning Environments, Virtual Learning Environments, Interoperability

1 Introduction

In the paper “Personal Learning Environments: Challenging the dominant design of educational systems”, Wilson et al. [1] describe the VLE (Virtual Learning Environment; also called a Learning Management System) as having become a dominant design for educational systems; this is a situation which limits choices for users and institutions, while also presenting a very challenging environment for an alternative design to gain any traction.

In this paper we describe an approach to challenging the dominant design through creatively subverting the VLE using highly interactive applications (Widgets) that can be delivered within the VLE but also embedded by the users into other platforms, including individually-owned tools and websites. By extending the capabilities of the VLE in this manner, we can create a new conversation about the VLE that moves us away from the dominant design, but stay within the comfort zone of lecturers, managers and students who have become used to the model. Also, rather than attempt to “create” a Personal Learning Environment (PLE) that is provided to learners, we instead open up the VLE to be remixed by users to construct their own PLE using technologies of their choosing.

The approach we took was to extend the Moodle VLE [2] using Wookie Widget Server (both open source), extended to support the Google Wave Gadgets API[3] and to provide embed code to let users take Widgets elsewhere. The experimental system was made publicly available for testing, and existing Moodle users were encouraged to try it.

2 Architecture & Design

2.1 Design Objectives

The objective for the experiment was to create an instance of Moodle that made use of the emerging W3C and Google Wave technologies in a way that revolutionizes the capabilities and use of the VLE platform, while at the same time requiring little or no modification of the VLE itself. In particular, we wished to deliver a very new kind of user experience based on synchronous, live-updating collaborative applications, and to enable these applications to be easily extracted by users so they could use them in personal platforms, such as Netvibes personal pages.

To realize this design we built on previous work [4, 5] and used Wookie, an open-source Widget engine, to render the Widgets and place them inside Moodle using iFrames (Figure 1). As well as providing a clear separation of concerns – allowing Widgets to be rendered for other platforms – this also provides a measure of security, as provided that Wookie and Moodle are served from different domains, any Widgets are unable to access information from within Moodle by accessing the data on the current page.

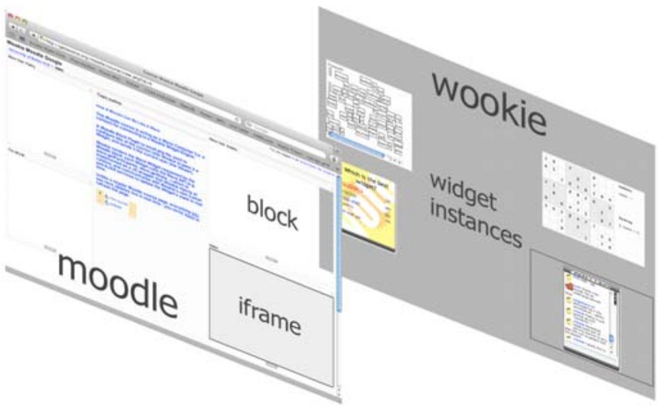


Figure 1: Relationship between Moodle and Wookie.

2.2 Standards

The key interoperability standards used were the W3C Widgets family of standards, and the Google Wave Gadget API. The **W3C Widgets Packaging and Configuration specification** [6] defines how Widgets are packaged, and their metadata. **W3C Widgets API and Events** [7] defines an API for per-instance persistence of preferences for Widgets, and access to metadata attributes. The **Google Wave Gadget API** [3] defines an API for families of Widget instances that share a common context, with methods to support shared state information and participants.[1]

Used together, this package of standards enables authors to create live-updating, collaborative applications that can be packaged and deployed in a standard fashion, with no dependency on a particular server platform or container application.

2.3 Components

The experiment involved the use of a standard Moodle distribution (version 1.9). The customization of this system involved two major components: the *Wookie Widget Server* and the *Wookie Moodle Block*.

The **Wookie Widget Server** is the component responsible for generating instances of Widgets, providing Widget APIs, and managing widget state persistence. For this experiment, the most significant implementation work was updating the Wookie server to also support the Google Wave Gadget API.

[1] The Google Wave platform consists of several different protocols and APIs; the Wave Gadget API is only a small subset of this platform. We did not implement the Wave Federation Protocol or Robots API. This means that, for example, Moodle does not become part of a Wave federation, and courses do not become wavelets; only the mode of interaction with widgets is made available.

The **Wookie Moodle Block** is a small PHP plugin for the Moodle VLE that uses the “blocks” extension mechanism offered by Moodle that allows course administrators to add content to course sidebars. The implementation did not make any changes to the core Moodle VLE at all, only using the Block extension system.

While Moodle is responsible for its usual core functions of managing courses and users, the *Wookie Moodle Block* is responsible for communicating with Wookie to request instances of Widgets, and then delegate rendering of applications to Widgets rendered by *Wookie Widget Server*. Running instances of Widgets then communicate with the *Wookie Widget Server’s* AJAX API and receive push events using its Comet service. This architecture is depicted in Figure 2.

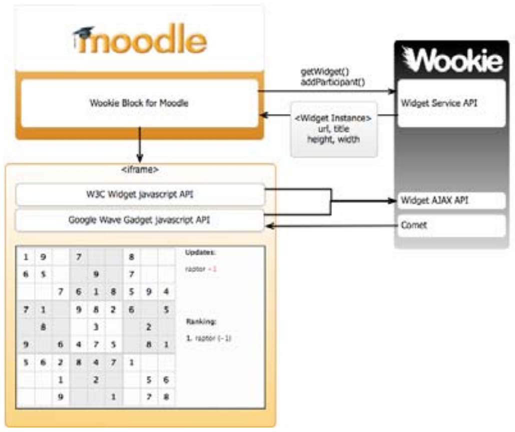


Figure 2: Detailed architecture

3 Implementation

3.1 Setup and configuration

We installed Moodle 1.9, Wookie Server, Tomcat, and MySQL on an Ubuntu server; this server was not dedicated to the experiment, but was instead a shared development server running a number of applications and websites. However, it performed very well under test.

3.2 Implementing the Wave Gadget API

It was relatively simple to map the Wave Gadget API onto our prior implementation of Widget shared state persistence (see [4,5]). We also used the

Moodle Block identifier as a way to generate a contextual identifier for managing shared states – the instance of the block within the course provides a common context for a shared state.

We were also able to create participants by passing user profile information to Wookie when instantiating Widgets. However, Moodle v1.9 does not have an events system we could use to remove participants, which meant that our test courses had large numbers of users, many of which were largely inactive. This was itself quite useful for testing purposes.

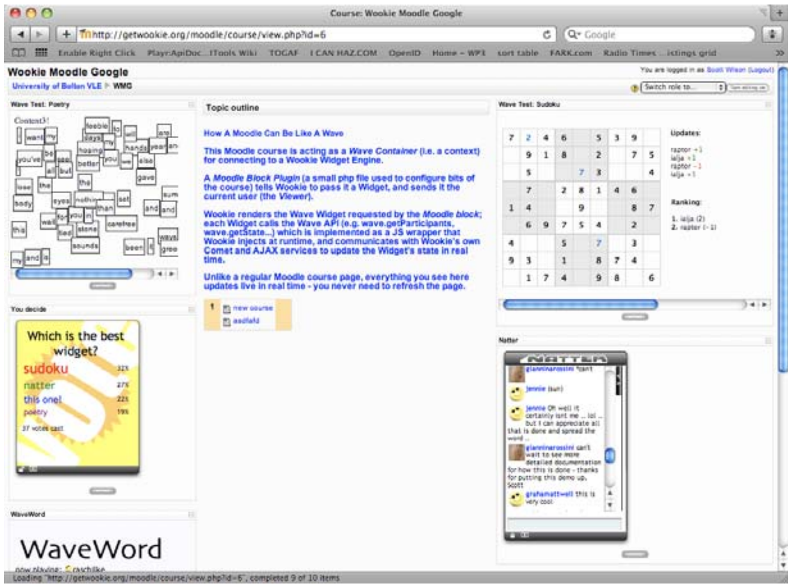
3.3 Widgets

For our prototype we converted the demonstration Google Wave Gadgets developed by Google[2] and installed them in Wookie. This involved minor changes to the way the applications were packaged, to make them conform to the W3C Widgets specification. Otherwise no major changes were required. The Widgets chosen were a Sudoku game and "fridge magnet poetry"; some other 'test' widgets were converted, but only to check the API implementation was working correctly. The Google Widgets, while not perhaps profound examples, are nevertheless useful in demonstrating the features of live-updating collaborative Widget applications.

In addition to the Google Widgets, we also developed two of our own Widgets using the Wave Gadget API. One of these was a chat/wall Widget called Natter (updated from a previous version that used our own shared state APIs), a voting Widget called You Decide, and a word-making game called WaveWord. Developing these Widgets was relatively straightforward.

The following screenshot (Figure 3) shows all of these Widgets in action within a single Moodle course.

[2] See http://code.google.com/apis/wave/samples/index.html



How A Moodle Can Be Like A Wave

This Moodle course is acting as a *Wave Container* (i.e. a context) for connecting to a Wookie Widget Engine.

A *Moodle Block Plugin* (a small php file used to configure bits of the course) tells Wookie to pass it a Widget, and sends it the current user (the *Viewer*).

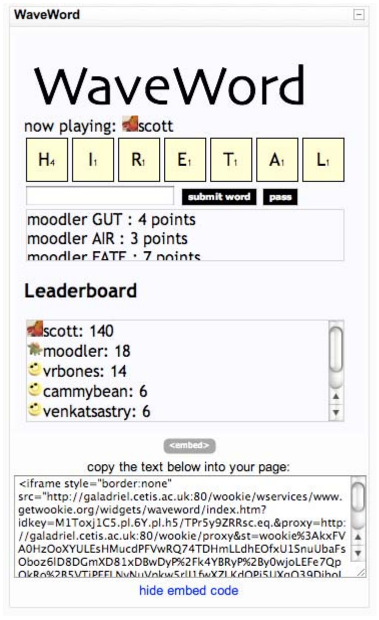
Wookie renders the Wave Widget requested by the *Moodle block*; each Widget calls the Wave API (e.g. wave.getParticipants, wave.getState...) which is implemented as a JS wrapper that Wookie injects at runtime, and communicates with Wookie's own Comet and AJAX services to update the Widget's state in real time.

Unlike a regular Moodle course page, everything you see here updates live in real time - you never need to refresh the page.

Figure 3: Screenshot showing live system

3.4 Embedding

As well as using the Widgets within the VLE, we also made it possible to take the Widgets, including all their contextual information, and embed them in other systems. Each Widget in the VLE also has a small "embed" icon underneath. Clicking this icon reveals a HTML code snippet that the user can place on another site (Figure 4).



moodler GUT : 4 points
moodler AIR : 3 points
moodler FATE : 7 points

scott: 140
moodler: 18
vrbones: 14
cammybean: 6
venkatsastry: 6

copy the text below into your page:

```
<iframe style="border:none"
src="http://galadriel.cetis.ac.uk:80/wookie/wservices/www.
getwookie.org/widgets/waveword/index.htm?
idkey=M1Toxj1C5.pl.6Y.pl.h5/TPr5y9ZRRsc.eq.&proxy=http:
//galadriel.cetis.ac.uk:80/wookie/proxy&st=wookie%3AkxFV
A0HzOoXYULEsHMucdPFVwRQ74TDHmLLdhEOfxU1SnuUbaFs
Oboz6lD8DGmXD81xDBwDyP%2Fk4YBRyP%2By0wjoLEFe7Qp
OkRo%2B5VTiPFFLNvNuVnkw5rlU1fwXZLKdOPi5UXqO39Dibol
```

Figure 4 - Example of embed code

3.5 Live testing

Once the experimental system was in place, we invited participants to register and try out the experience. On June 12th, 2009, we announced the system in a blog post[8], on Twitter, in the Moodle developer forum, and on the Google Wave developer forum. Within a few hours we had acquired quite a few users. Initial issues with email-based registration meant giving out a default account for guests, and so we don’t have exact figures on the number of users. However, we estimate that around 200 users tested the system in the initial 7 days (12th-19th June); these comprised 124 named accounts, and at least 90 distinct IP addresses using the “moodler” public test user.

We were pleased to note that despite the system being open for public test, there were no performance, security or stability issues with the system, which ran with only a single restart, which was to update the Wookie code to a newer release. Users noted no major problems; one user reported an AJAX error, probably caused by a dropped connection, for one of the Widgets on one occasion.

Overall the performance results were very promising, as the prototype itself was running on a shared development server (running several other applications), with no special optimization in place (e.g. level two cacheing was not enabled for the database, and there was no special tweaking of our webserver for better Comet performance). This gives us confidence that a large-scale demonstrator could be attempted to explore the performance boundaries of the system.

The Natter chat Widget also provided a means for us to capture comments from testers. The reaction was very positive; below is a small selection of comments received from testers.

- Very cool. I've got two browers open and see my poetry changes live!!
- This is fun!
- sat here in schiphol - if they get this wave tech right it will be incredible
- I like this effort
- thanks for writing poetry with me
- It is very interesting I am gonna read this weekend some articles I took from Waveprotocol.org Just to understand the way it works
- I think wave is phenomenal! Can't wait ...
- This is a very nice demonstration of Google Wave and integrated into Moodle. Well done guys!
- great stuff!
- This looks like it could be useful - need to get our VLE team on it
- this looks great much less static that the usual moodle stuff
- can't wait to see more detailed documentation for how this is done - thanks for putting this demo up,
- good work … Moodleusers luv ya baby!!!
- how cool!
- very inspiring!
- I love it...can anybody help me to do this? add wave in moodle?
- This looks good!
- is there a whiteboard app?
- do you know where I can download the source?
- Writing from the natter widget, embedded in my blog. Nice the way the highlights push it off the page. Now, let's see it appear over on the Moodle site...ah, here it is ☺

4 Analysis

What does it mean to have a “Moodle Wave”? While the technical work on integration did not bring about any particular surprises, the end result is a very different user experience from the conventional Moodle VLE.

4.1 Replacing VLE functionality

It was expected by the team that the use of Widgets, particularly those using the Wave Gadget API to provide live collaboration, would be a satisfactory replacement for the built-in tools of the VLE, such as chats, forums, and quizzes. This was certainly the case; and we believe that almost the entire toolset of a VLE can be readily replaced with this type of Widget. Martin Dougiamas, the original developer of Moodle, made the following comment after the Google Wave video was released by Google[9]:

"Moodle (and similar systems) are (like it or not!) more about the *management* of learning at a higher level than this, a system of control and authentication that supports institutional policy and practice.

However, what I do think will happen is that many of the activities in Moodle will be replaced by services like these" [10]

This has significant implications for the VLE. If the majority of the features of the platform are readily outsourced to – possibly superior – external tools, then this requires a rethink of what defines a VLE.

For example, a version of Moodle that was developed with the assumption that it would be connected up with a Widget server would presumably not have large numbers of pre-developed PHP modules, but instead focus on areas outside the concern of individual Widgets, such as course and user management, integration and reporting to institutional systems such as the student record system, and perhaps integrating tools for structuring activities around Widgets, such as LAMS[11].

Would this more limited product profile make it easier for developers to create new, perhaps more innovative VLE-type products, at a more affordable price? Or would it instead make a case for bypassing standalone VLEs and instead integrate Widgets into more generic systems such as Sharepoint?

A counter-argument could be made that the existence of "cut and paste" embedding of widgets via JavaScript into the VLE is already possible, and has not had such an effect. However, there are considerable differences in the affordances of this approach versus the one we describe here. However a discussion of this point is beyond the scope of this paper.

4.2 New modes of interaction

Rather than just being a like-for-like replacement, the Widgets update in real-time without any page refresh, and this affects user behavior. Rather than clicking links to launch tools or to view content, the Widgets encourage more of a "monitoring" mode of operation, with users navigating to a course page, then leaving it open in the background, occasionally bringing it into focus to see if any new conversations were happening in the chat widget, or the voting results had changed.

This was visible in the activity reports for Moodle, which is conventionally used to see student activity. Rather than the usual histogram of page views seen in live courses, this showed a sparse distribution of visits to course pages. This may be an artifact of the type of informal testing we conducted, and so we identify this a hypothesis for testing in future pilots using the system.

4.3 Micro-tracking

Another consequence of Widgets is that far less tracking information is available at a micro-level, as interactions with Widgets are not made available to the VLE. This implies that a new model for tracking will need to be developed for such systems; this may be a useful opportunity to reconsider tracking in more sophisticated terms than page views and hits. For example, it may be useful to separate out measures of attention, using something like APML[12], and measures of user effort, using something like User Labor Markup Language (ULML[13]).

4.4 Embedding – the fuzzy boundary between institutional and personal systems

While the ability to embed Widgets from the VLE into other systems was not a significant aspect of this experiment, it was noted that several people had embedded Widgets from the example course into blog posts. One of the issues with embedding is that what for some people may be a "private" space could very easily spread into the public domain if participants embed Widgets from a course into public sites. The current architecture offers no practical means of enforcing restrictions on embedding, and so this would require policies and user education. A related issue is that it is hard to track where Widgets are being embedded; this could be addressed by placing tracking information into generated embed codes, which may also offer a solution to unauthorized public sharing. Striking an effective balance between enabling individuals to use PLEs, and supporting the rights of others will be one of the key challenges to be addressed.

5 Conclusions and Future Work

Our experiment demonstrates that it is feasible to embed live-updating collaborative applications within a VLE that can also be easily shared and embedded in personal technologies, and that this can be done using open standards and specifications. The Wookie technology supporting this experiment has been shown to be robust enough to support open testing, and is ready for a larger-scale demonstrator.

However the challenges raised by this work are significant and will need to be addressed. In particular the tracking and monitoring functionality in VLEs – often posited as one of the key features of such systems – may be rendered partially or completely ineffective by Widgets. This should be examined in future studies, as if it is shown to be the case, existing tracking systems will need to be redesigned to cope.

We believe the approach taken in this work is going to be compelling for both students and teachers alike, and that this will challenge assumptions of what a "VLE" is and its relationship with other technologies. Eventually this could break down the dominant design of VLEs and open up institutions to more innovation in education technology.

References

Acknowledgments. The work on this publication 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

[1] Wilson, S., Liber, O., Johnson, M., Beauvoir, P., Sharples, P., and Milligan, C. (2007). Personal Learning Environments: Challenging the Dominant Design of Educational Systems. Journal of e-Learning and Knowledge Society 3(2), 27-38.

[2] Moodle.org: open source community-based tools for learning. http://www.moodle.org

[3] Wave Gadgets API Reference. Retrieved from http://code.google.com/apis/wave/extensions/gadgets/reference.html on 20th June, 2009.

[4] Wilson, S., Sharples, P., and Griffiths, D. Extending IMS Learning Design services using Widgets: Initial findings and proposed architecture. in Current Research on IMS Learning Design and Lifelong Competence Development Infrastructures 2007. Barcelona: Universitat Pompeu Fabra, Grup de Tecnologies Interactives.

[5] Wilson, S., Sharples, P., and Griffiths, D. Distributing education services to personal and institutional systems using Widgets. Workshop on Mash-Up Personal Learning Environments (MUPPLE'08) at the 3rd European Conference on Technology Enhanced Learning (EC-TEL08). 2008. Maastricht.

[6] Widgets 1.0: Packaging and Configuration. W3C Working Draft, 28 May 2009. Retrieved from http://www.w3.org/TR/widgets/ on 20th June, 2009

[7] Widgets 1.0: API and Events. W3C Working Draft, 23 April 2009. Retrieved from http://www.w3.org/TR/widgets-apis/ on 20th June, 2009.

[8] http://www.cetis.ac.uk/members/scott/blogview?entry=20090612190435

[9] Google Wave Developer Preview (video), Google I/O Conference 2009. Retrieved from http://wave.google.com/ on 20th June, 2009.

[10] Dougiamas, M. (2009). Comment on Moodle Developer Forum. Retrieved from: http://moodle.org/mod/forum/discuss.php?d=124670&parent=548854 on 20th June 2009.

[11] Learning Activity Management System; LAMS Foundation, retrieved from http://www.lamsfoundation.org/ on 20th June, 2009.

[12] Attention Profiling Markup Language. Retrieved from http://www.apml.org/ on 20th June, 2009.

[13] Arikan, B., and Erdogan, E. (2008). User Labor: A framework for sustaining user labor across the web. Retrieved from http://userlabor.org on 20th June, 2009.