Moodbile: a Moodle web services extension for mobile applications

Jordi Piguillem, Marc Alier, María José Casany, Enric Mayol, Nikolas Galanis

Services and Information Systems Department, Universitat Politècnica de Catalunya BarcelonaTech, jpiguillem, ludo, mjcasany, mayol, ngalanis}@essi.upc.edu

Franciso J. García-Peñalvo, Miguel Ángel Conde

USal Universidad de Salamanca, {fgarcia, mconde}@usal.es

Abstract

The Moodle 2.0 Web Services architecture has been designed to optimize bulk operations or administrative tasks. Therefore, it does not support external applications accessing activity modules. For this reason, the Moodbile project aims to provide an extension of the Moodle 2.0 Web Services architecture in order to provide access to external applications such as m-learning applications, from within Moodle.. To validate this extension, three m-learning applications have been developed: an HTML5 client that may be used from most mobile browsers, an android-native app and an iOS-native app.

Keywords

E-learning, m-learning, web services, integration, Moodle, mobile apps.

1. Introduction

Over the last 10 years, the Moodle community has shown a huge growth. Moodle.org proudly shows updated statistics of community members, servers installed and other data about the registered sites, which is just a subset of all the Moodle installations out there. The number of Moodle's usage scenarios is only limited by the imagination of every teacher using Moodle. Moodle.org hosts hundreds of third part plug-ins (activity modules, courses, blocks, themes, integrations and hacks), and the contribution base has been growing so much that a new database and quality assurance procedures had to be set in place.

The evolution of the official Moodle distribution is (and needs to be) determined by the decisions of Martin Dougiamas, who has to care for the product quality, relevance in the state of the art, wishes of the community and sustainability of Moodle. With each new release, decisions that affect the presence or absence of features, capabilities, interfaces etc. are made. And while the result is good enough for a large base of users, third party contributions show up with requirements that are not fulfilled by the standard distribution.

The M of Moodle stands for Modular because in its earliest version Moodle was already designed to be extended with activity Modules, Blocks, Lang extensions, course formats etc. But non-standard plugins are not always a clear shot. Moodle extension developers have their work cut out for them trying to assure the quality of their code and achieve compliance with every possible Moodle setup while maintaining compatibility with the new Moodle versions. An extension developer must take into account all upcoming upgrades and bug fixes, ensuring stability and data integrity. Despite that, a lot of members of the community continue creating, maintaining and using extensions (Cole & Foster, 2008).

This paper is about the research our group has been doing the last 5 years in the problem of enabling Web Services for Moodle. Section two summarizes previous work we participate to extend Moodle functionality by means of external application integration. Section three describes the Moodbile project putting special attention to the extension we propose to the Moodle Web Service Architecture. Finally, we conclude with the main conclusions and further work.

2. Related Work

This section summarizes the main LMS's services and functionalities that some initiatives propose to extend the scope of the LMS to the world of mobile devices.

- Lehner and Nosekable (2002) did one of the first studies about mobile devices interacting with virtual campuses. The Welcome system offers access to certain contents and services (such as calendars or events) of the virtual campus using mobile devices, and interacting by means of SMS messages.
- Triffonova and Ronchetti (2004) and Colazzo, Molinari, Ronchetti and Trifonova (2003) identified LMS's services and functionalities that may more relevant to be accessed from a mobile device. They also identify the main issues of a LMS's architecture to be considered from a mobile device. These architectural issues are: 1) context discovery (the system must check automatically the mobile device features and decide which services may be provided), 2) adaptation of contents, and 3) synchronization between the mobile device and the LMS. They have developed a mobile prototype based on this architecture.
- Hinkelman (2005) developed a Moodle 1.6 module to do testing using mobile devices. This version mainly offered testing services and feedback to students. Due to technological issues, this project was developed to work with Japanese mobile phones (because the tool is based on CHTML and 98% of the Japanese mobile phones supported this language). Afterwards, (Cheung et al. 2006) presented a study to adapt Moodle to mobile devices centred in the adaptation of contents.
- The Open University has been working on Moodle extensions to mobile devices for quite long time..At 2009, they proposed a Mobile VLE for Moodle, a m-learning application to access Moodle from mobile devices. This application provides a subset of Moodle functionalities selected by popular polls to students (mobile users). These LMS's functionalities were: assessment scores, messages (read course messages and unread forum posts), tasks ('tick-boxes' to see the progress of course activity), tasks planning and resources (read and download if it is supported) (Thomas, 2010).
- Momo (Mobile Moodle) and MLE (Mobile Learning Engine) projects developed m-learning applications to access some Moodle 1.9 functions (Momo, 2008), (MLE, 2009). Both m-learning applications are is based on J2ME, while the MLE project developed a client application and an additional web version to access Moodle courses from mobile browsers. Some of the Moodle modules/activities supported by this project are: lesson, quiz, task, resource, forum, survey, choice, wiki (read only), database (search and query) and message.
- Project MPage develops a Moodle 1.9 client for iPhone (MPage, n.d.). Some of the Moodle modules and activities supported by this project are: view course categories, access MyMoodle, edit events, access to resources in different formats, chat, choice, forum and quiz.
- The MLE and Mpage projects are designed as Moodle hacks. Both systems offer their main functionality using Moodle blocks. MLE re-implements the logic of some Moodle services to be offered to mobile web clients using XML and CSS mobile. The Mpage system offers two proprietary mobile applications that access the Moodle data using a proprietary web service. It rewrites part of the Moodle code to offer its functionality to mobiles.
- Moodle.org has identified main functionalities for an iPhone client for Moodle: 1) to upload video, audio and other file formats to the user's private space in the Moodle server. 2) to view users enrolled in courses where the user also be. 3) to view and download activities and content of a course. 4) to view student grades, to make grade assignments and to download them. 5) to receive notifications from the Moodle server and send internal email. 6) to view forums, discussions and to create and reply posts. 7) to view calendar events and assignments deadlines. However, the current version of the prototype designed by Moodle.org only allows uploading files to the user's private space in the Moodle server, view course participants and view the list of activities and contents of a course.

3. Previous work

3.1 The Campus Project

The Campus Project (http://www.campusproject.org), was the initiative of several Catalan universities to come together and create a virtual open source campus infrastructure. The Campus project had to bind in the same Virtual Learning Environment (VLE) up to 23 different educational existing applications developed by the project partners. The open source LMSs used in the campus project were Moodle and Sakai (Santanach et al. 2007) The purpose was to be able to launch external applications from the LMS providing them with the basic user authentication and authorization information. Besides, the external application had to provide logging information (i.e students activity) to the LMS.

The Campus Project was designed as an integration platform of e-learning applications and was based on a service-oriented architecture (SOA). Part of the architecture consisted in the creation of a web service layer for Moodle so that it could provide external applications with authorization and authentication information. The OKI (Open Knowledge Initiative) OSIDs (OKI, n.d.), IMS TI (IMS TI 2012) were used as framework to integrate these applications in the LMS.

3.2 The Moodle 2.0 Web Services Architecture

Moodle is composed of the Moodle Core and several subsystems around it. Moodle Core includes the basic functionality of the learning platform (users, courses, groups, etc) and since Moodle 2.0 these functionalities are offered as a more structured API (Application Programming Interface). The other subsystems implement specialized functionalities and features such as activities, repositories, filters, blog, messaging or tags.

Based on the experience of the Campus project, in 2008, the SUSHITOS research group designed a solution to enable Moodle to provide some of its functionalities using web services. This solution was the Moodle Web Services Architecture, which has been implemented for Moodle 2.0 and released in late 2009 (Alier et al. 2010a), (Alier et al. 2009). Moodle 2.0 provides tools to extend web services in a standard way so that third-part web services could be added to the system.

The Moodle Web Services Architecture adds two logical layers to Moodle's architecture (shown in Figure 1). The first one, called Moodle External API Layer includes the logic of each service. The second one is the Web Services Connectors Layer. The Moodle Web Services Architecture is not bound to a specific web services protocol; it is designed to be protocol independent. For each supported protocol (SOAP, REST, XML-RPC and AMF) there is a specific web services connector module in this layer. Each web services connector implements the translation of the methods implemented in the Moodle External API to the specific protocol and syntax. The Web Services Connectors layer is an extendible layer that allows the addition of new communication protocols.

A key element in the design of the Moodle Web Services Architecture is its extendibility based on plug-ins. The Moodle External API can be extended in a safe way, giving full security control to the Moodle administrator. If a new service or a new kind of web services protocol is needed, a developer may define and implement it as an extension to those provided by Moodle.

3.3 Projects using the Moodle Web Services Architecture

Several research projects in which the Moodle Web Services architecture was used have been proposed recently. In each project, additional services had been added to the external layer, and new custom web services connectors to support specific web services and authentication protocols. For example, Layers4Moodle project allows the students to create annotations and embed content over Moodle and share it with the rest of the course in a social stream, using OAuth authentication. Moreover, the TRAILER project uses a Moodle activity module that gathers informal learning activities through web services.

3.4 The Moodbile Motivation

Moodle 2.0 shipped with the Moodle Web Services Architecture but providing a limited amount of services and connectors, enabling mostly web services and security strategies for server administration and bulk actions. However, it came with very limited support for accessing information from activity modules, contents and other components required to implement a Mobile client. Moodle HQ has released an official Moodle app for iOS and Android, but their functionalities are very few and the set of services are limited.

There are several proposals trying to bring Moodle to the Mobile-Learning space. Moodle needs more than the official Moodle mobile application, it is necessary more research and developer groups working in this direction, as for example, exchanging ideas at the "Using Moodle" course and the "Moodle for Mobile" forum (www.moodle.org). This kind of research and development activity will benefit the Moodle community.

With the Moodbile project we wanted to aggregate all the developments on Moodle web services extensions, documentation and mobile clients, providing a stable API of web services, protocols and authentication methods that will be independent of the underlying Moodle version. This API aims to be a valuable contribution for the Moodle community and a platform where other groups can build their own mobile clients.

4. Moodbile Project

4.1 Moodbile Research Goals and Requirements

Our goal was not to create a web service layer to access every single Moodle activity feature, but to design an extension of the Moodle Web Services Architecture that provides access to the most suitable Moodle features for mobile applications. To gather these features, a log analysis of the Moodle server of our university (UPC)

has been performed. This analysis considered two data sources: the web server logs and the Moodle log. From both sources, we have been analysing which activity modules and actions performed on the Moodle platform where used more often from mobile devices (Casany et al. 2012). An additional data source for requirements has been the previous projects described in section 2.

The Moodbile extension has been designed considering user needs instead of optimizing bulk operations or administrative tasks. This is why this extension does not follow many of the premises used to develop the Moodle Web Services Architecture. Besides, the Moodle Web Services Architecture provides only basic services such as course enrolment, group and user management, etc. It does not provide services to access additional Moodle features such as activity modules or other plugins.

4.2 Moodbile Architecture

This section describes the proposed extensions to the Moodle Web Services Architecture to provide additional web services for mobile applications (Casany et al. 2012). This involves extending the three layers as shown in Figure 1: the Core extension re-implements some features that Moodle does not provide in the proper way to be used by web services. The External API layer is where the actual services for mobile integration are defined and implemented. The Connectors layer is used to provide additional web service protocols. Moodbile also includes some new authentication plug-ins, such as OAuth (OAuth, 2010) (more suitable for mobile devices), some Moodle blocks and new administration and user panels, among other features and improvements

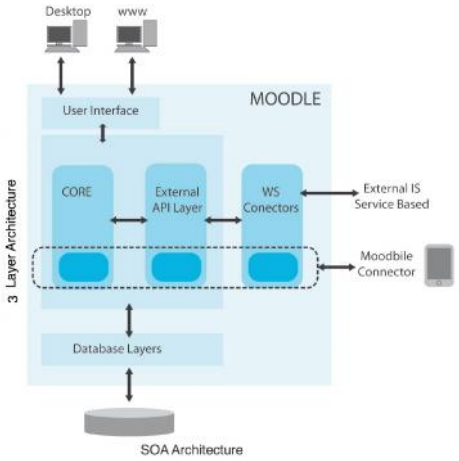

Figure 1. Moodbile architecture

The External Layer extension is implemented as a Local Moodle plug-in avoiding collision with Moodle External files and other extensions. The rest of the features are implemented through the extension mechanisms offered by Moodle. Our extension is based in the following additions:

New authentication method.

Moodle Web Services are oriented to academic management and bulk operations such as course creation, new user registration and student enrolment. Moreover, Moodle services consumers are under control of the same organization, so complex authentication methods are not required. Moodle users authenticate with their username/passwords or with special access tokens. To secure the connection, it may be enough to restrict IPs that will be able to invoke this web service.

In a mobile environment, such IP restriction mechanisms are not enough, since user/password or token authentication is not secure since this information is sent in each web service invocation from a mobile device, and it is not possible to establish a white list of authorized IPs. To solve this problem, Moodbile project applies an OAuth authentication mechanism (OAuth, 2010) allowing teachers and students to access Moodle using third-party applications in a secure way.

New Connectors.

Moodle offers several web service protocols but some of them are not suitable for a mobile environment. For this reason, Moodbile proposes several new connectors. To give support to HTML 5 Moodle extensions, an AJAX compatible connector with JSON messages was implemented. This connector uses the Moodle session, so the user must be logged in to use an HTML 5 client. In a similar way, JSONP allows HTML 5 external applications to access Moodle web services using OAuth authentication. Moreover, JSON-RPC connector was implemented to add a lightweight protocol that can be used effectively from a mobile phone. This connector uses Moodle web services authentication methods. Finally, another JSON-RPC connector and a new REST connector were implemented using OAuth authentication. These connectors are the most commonly used for Moodbile clients for their ease of use and their security improvements.

New Services.

New services are defined into the External layer in several components. Components are classified into Basic Services, Course Content Services, Personal Content Services or Platform Services.

- Basic Services. Course component provides basic services to retrieve user course information. User component provides services to list course enrolled users and their profiles. Group component gives services access to groups and their group members.
- Course content services. Services in this group allow the interaction with course content and activities.
- Thus, Assignment component provides services to get assignment information and to submit homework. Forum component facilitates management of forum information, discussions and posts. Resource component provides all the services related to digital resource access. Quiz component services are used to export quizzes to XML and QTI formats.
- Personal content service. Services in this group provide access to users' personal content. Blog component allows read, write and access to Moodle blogs. Calendar component gives access to user calendar, scheduling and events. Grade component allows retrieving users grades and outcomes. Message component allows using the internal messaging system.
- System Services. Lang component provides I18N services to develop clients that use the same terminology of the host. File component allows the upload and download of files from Moodle in a secure way. And finally, System component provides host and system information services.

Moodbile Clients.

Three mobile clients have been developed to test Moodbile architecture and services, and to conduct research in m-Learning topics. These client's interface are shown in Figure 2.

Figure 2: Moodbile client apps: Html5, Android and iOS Clients.

- Moodbile HTML 5 Client. It is a Moodbile client that runs on most mobile browsers. Moodbile HTML 5 Client runs on the same web server as the LMS and provides a mobile web friendly limited front, using the Moodbile web services specification.
- Moodbile Android Client. This client is an Android App that provides a limited front end to the LMS written on native Android code. The android-native application supports online and offline work when network coverage is not present or expensive. This application has a persistent data storage for the contents retrieved from the Moodle server. When working offline this feature allows a quick access to the local contents using a memory cache. This app can also connect to different Moodle servers using two different authentication protocols (OAuth or username and password). Synchronization is still under development.

- Moodbile iOS Client. This client is an iOS app that provides a limited front end to the LMs written on native Objective C code. Like the Android client this app supports online and offline work when networked coverage is not present or expensive. This application has a persistent data storage for the contents retrieved from the Moodle server. When working offline this feature allows a quick access to the local contents using a memory cache.

Moodbile Spec.

The Moodbile Spec is the result of a service modelling and engineering process. SUSHITOS Research Group has designed and optimized Moodbile Web Services with the objective to give specialized support to external applications. Moreover, this engineering process also takes into account how Moodbile objects are modeled. Moodbile objects are designed using the respective Moodle ones as a starting point to obtain new objects independent of Moodle internal features. This process has resulted in a web services API that can be used and understood by a non-Moodle programmer. The detailed description and specification of Moodbile services is shown at http://docs.moodbile.org/index.php?title=Moodbile_WS_Latest_Version_Documentation.

4.3 Service definition

External Layer Design

One of the requirements of Moodle Web Services was to support a variety of web service protocols. To decouple External Layer from connectors, a protocol independent service description mechanism was designed.

```
class moodle_group_external extends external_api {
    public static function get_groups($groupids) {
        ....
    }
    public static function get_groups_parameters() {
        return new external_function_parameters(array(
             'groupids' => new external_multiple_structure(
                   new external_value(PARAM_INT, 'Group ID')
                ,'List of group id. A group id is an integer.'),
        ));
    }
    public static function get_groups_returns() {
        return new external_multiple_structure(
            new external_single_structure(array(
               'id' => new external_value(PARAM_INT, 'group record id'),
               'courseid' => new external_value(PARAM_INT, 'id of course'),
               'name' => new external_value(PARAM_TEXT, 'course unique name'),
               'description' => new external_value(PARAM_RAW, 'group description text'),
               'enrolmentkey' => new external_value(PARAM_RAW, 'enrol password'),
        )));
    }
}
```

Figure 3: Moodle External API example (Moodle_group class).

Every Moodle component that wants to provide any service must implement a componentname_external class and place it in a file called externallib.php. This class will be responsible for implementing and describing as many services as the component wants to provide. Every service provided by the External Layer is described and implemented by three methods. The service itself is implemented in a first method. A second method describes the service parameters and a third one describes the service return. By convention, if a service is implemented by the servicename method, the method that describes service parameters must be the servicename_parameters method and, the method that describes service returns must be the servicename_returns method. In Figure 3, we show an example of a service description from the Moodle_group component. With this technique, protocol connectors may provide and access services of the External Layer.

```
public static function get_group_by_groupid($groupid) {
    ...
}

public static function get_group_by_groupid_parameters() {
    return new external_function_parameters(
        array(
            'groupid' => new external_value(PARAM_INT, 'group id',
                 VALUE_REQUIRED, 0, NULL_NOT_ALLOWED),
        )
    );
}

public static function get_group_by_groupid_returns() {
    return Group::get_class_structure();
}
```

Figure 4: Moodbile External API example.

An important difference between our proposal and the Moodle definition of services by means of classes is the existence of the static method get_class_structure. This method is defined to support the service definition so that it is more reusable and maintainable, since it permits to define in a more transparent way complex parameters and returns of Moodbile services. In Figure 4, we show how to use it.

New web services connectors and authentication

As we have mentioned before, Moodle Services are designed for academic management and do not require to apply complex authentication methods. However, in a mobile environment, security issues are more critical and new protocols and authentication methods are necessary. To solve this problem, we have proposed and developed an OAuth authentication plug-in, as a part of Moodbile project, to be used in combination with web service connectors. Our intent was to not modify Moodle's internal code. Therefore, we have designed an extension of the Moodle connector architecture as it is shown in Figure 5.

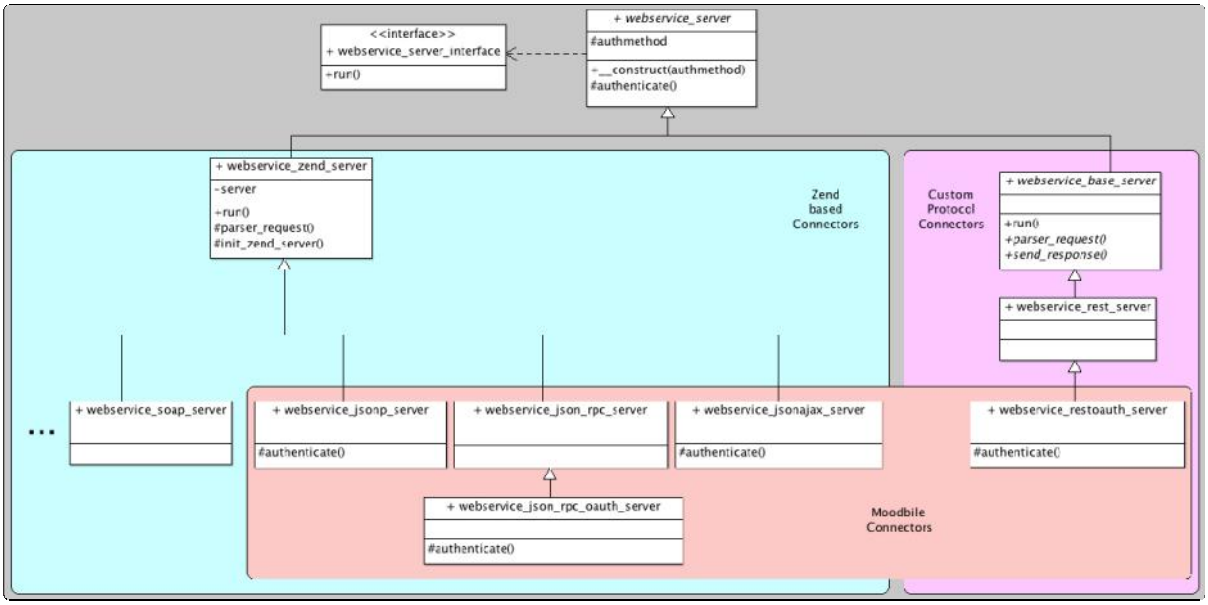

Figure 5. Moodle connector architecture

Moodle architecture proposes the webservice_server_interface to allow the implementation of two types of web service protocol connectors: Zend-based connectors and Moodle custom connectors. Connectors for XML-RPC, SOAP and AMF protocols delegate the protocol management to the implementation that offers the Zend Framework (http://framework.zend.com/), meanwhile the REST connector is a custom connector implemented by Moodle.

Our proposal has been to use the abstract class webservice_server that all Moodle connectors inherit to define new web service protocols. Moreover, its authenticate() method implements the logic to authenticate users using either username-password or token methods, which are the traditional authentication methods supported by Moodle. However, taking advantage that this method is defined with a protected visibility, we propose to redefine and implement additional authentication methods like, for example, OAuth.

The Moodbile project offers several new connectors, but only some of them can be used in combination with the OAuth authentication. The first one is a REST/OAUTH connector that extends the Moodle REST connector re-implementing the authenticate() method to support OAuth. The second connector is a JSON-RPC/OAUTH connector. Notice that in this case, this connector is a specialization of the JSON-RPC to which delegates part of the protocol logic definition, but it re-implements the authenticate() method to support OAuth. The JSON-RPC protocol inherits traditional Moodle authenticate methods. The new implementation of these authenticate() methods and of other scripts use OAuth-php libraries (http://code.google.com/p/oauth-php/) with some minor modifications to fully integrate with Moodle and Moodbile. Like almost all Moodle connectors, all JSON based connectors are based in Zend Framework.

The new protocols that the Moodbile project supports are used in different client implementations. For example, our Moodbile Android Client uses indistinctly the JSON-RPC and JSON-RPC/OAUTH protocols and our Moodbile HTML5 Client uses the JSON-AJAX protocol. In this latter case, the JSON-AJAX uses an authentication method based on the Moodle session. Finally, the custom REST/OAUTH protocol is used in a pilot we use to test our Moodbile web services. Some of these protocols are also used to integrate external applications. For example, the Layers4Moodle project uses JSON-P with OAuth authentication for its Chrome version, while its HTML5 Moodle plug-in uses the JSON-AJAX protocol.

5. Conclusions

The Moodbile project provides an extension to the Moodle web services designed to build mobile clients. We provide an HTML5 Mobile client and Android clients as reference implementation that will be used in learning experiences in the next academic course. Moreover, we see the best value of the Moodbile project in the Moodbile Spec, a web services API designed around the requirements of several projects that can be used as a platform to connect external applications, specially mobile ones, with Moodle.

The current implementation of the Moodbile spec does not yet comply with the necessary coding standards for it to be included in the Moodle core. This is an aspect to be improved and we hope the community can help here. We plan to start building new features such as Quiz support for the Moodbile clients to keep the Moodbile Spec API growing. As further work we also plan to test the clients and validate the Moodbile architecture by designing several pilots for students and teachers.

References

Alier, M., Casany, M. J., & Piguillem, J. (2009). Towards Mobile Learning Applications Integration with Learning Management Systems. Multiplatform e-learning systems and technologies: mobile devices for ubiquitous ICT-based education, 182.

Alier, M., Casany, M. J., Conde, M. Á., & García-Peñalvo, F. J. (2010). Interoperability for LMS: the missing piece to become the common place for e-learning innovation. International Journal of Knowledge and Learning, 6(2), 130–141.

Alier, M., Pedro, X. D., Casañ, M. J., Piguillem, J., & Galanis, N. (2010). Requirements for Successful Wikis in Collaborative Educational Scenarios. Intern. Journal of Knowledge Society Research (IJKSR), 1(3), 44–58.

Casany, M.J., Alier, M., Mayol, E., Piguillem, J., Galanis, N., Conde, M.A. and García-Peñalvo F. J. (2012): Moodbile: A Framework to integrate m-learning applications with the LMS. In Journal of Research and Practice in Information Technology (2012) to appear.

Casany, M.J., Alier, Mayol, E. (2012): Using LMS activity logs to measure mobile access. In Proceedings of third International Conference of Technology Enhaced Learning 2012, to appear.

Cheung, B., Steward, B. and McGreal R. (2006). Going mobile with Moodle: First steps. In Proc. of IADIS International Conference Mobile Learning. Dublin: International Association for the Development of the Information Society.

Colazzo, L., Molinari, A., Ronchetti, M. and Trifonova, A. (2003). Towards a multi-vendor mobile learning management system. In Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-Media), Hawaii, pp. 121-127.

Cosme, C.A., Pedrero, A. and Alonso, V. (n.d.). Moviltest: adaptación de cuestionarios de Moodle para dispositivos móviles. Available at: http://www.web.upsa.es/spdece08/contribuciones/177_moviltest.pdf

Cole, J & Foster, H. (2008). Using Moodle. O'Really Community PRESS Hinkelman, D. (2005). Moodle for Mobiles Project. Available at: http://moodle.org/mod/forum/discuss.php?d=33033

IMS LTI. (2012). IMS GLC Learning Tools Interoperability Implementation Guide. Available at: http://www.imsglobal.org/LTI/v1p1/ltiIMGv1p1.html.

Lonn, S. and Teasley, S.D. (2009). Saving time or innovating practice: Investigating perceptions and uses of Learning Management Systems. Computers & Education, 53(3), pp. 686-694.

MLE. (2009). Mobile Learning Engine Project. Available at: http://mle.sourceforge.net/

Momo. (2008). Mobile Moodle Project. Available at: http://www.mobilemoodle.org/momo18/ MPage. (n.d).

MPage Project. Available at: http://massmedia.hk/moodle/course/view.php?id=2

OKI n.d. First PHP O.K.I. Summit. Available at: http://sourceforge.net/apps/mediawiki/harmoni/index.php?title=First_PHP_O.K.I._Summit.

Oauth (2010). The OAuth 1.0 Protocol RfC. Internet Engineering Task Force (IETF). 2010. EHammer-Lahav E. Editors. available at: http://tools.ietf.org/html/rfc5849

Santanach, F., F. S., Dalmau, J. C., Casado, P. C., & Alier, M., M. A. (2007). Proyecto CAMPUS. Una plataforma de integración. e IV Simposio Pluridisciplinar sobre Diseño, Evaluación y Desarrollo de Contenidos Educativos Reutilizables. SPDECE 07.

Thomas, R.C. (2010). Mobilizing the Open University: case studies in strategic mobile development. Journal of the Research Center for Educational Technology, 6(1), pp. 103-110.

Trifonova, A. and Ronchetti, M. (2004). A general architecture to support mobility in learning. In Proc. of IEEE International Conference on Advanced Learning Technologies (ICALT), Joensuu (Finland), pp. 26-30.

Acknowledgements

This work has been funded by the “Spanish Ministry of Science and Innovation” in project MiPLE code TIN2010-21695-C02-02.8 and the TRAILER project funded by the European commission (http://grial.usal.es/agora/trailerproject).