COMPARISON OF THE MAIN ALTERNATIVES TO THE INTEGRATION OF EXTERNAL TOOLS IN DIFFERENT PLATFORMS

Carlos Alario-Hoyos[1], Scott Wilson[2]

[1]School of Telecommunication Engineering, University of Valladolid,
Paseo de Belén 15, 47011, Valladolid (SPAIN)
[2]Institute for Educational Cybernetics, University of Bolton,
Deane Road, BL3 5AB, Bolton (U.K.)

Abstract

The integration of third-party external tools in Virtual Learning Environments (VLEs) and social networks enhances the flexibility and the ease of customisation of successful systems such as Moodle and LAMS. In this context, several authors and organisations have proposed different ways to deal with the integration problem. Some of the most representative approaches are: IMS Learning Tools Interoperability (IMS LTI), Basic LTI, the Apache Wookie (Incubating), and the Group Learning Uniform Environment (GLUE!) architecture. This paper analyses the main features of each of these approaches, such as the number and diversity of tools that can be integrated, the degree of coupling involved, and the richness in the communication, with the aim of establishing a comparison that may help to decide the adoption of some of them, depending on the expected requirements.

Keywords: *Learning platforms*, *integration*, *external tools*, and *contracts*.

1 INTRODUCTION

The use of Virtual Learning Environments (VLEs) and social networks in education such as Moodle[1], LAMS[2], Sakai[3] or Elgg[4] has become more and more popular in the last few years due to the emerging interest in distant and blended learning [1]. Nevertheless, the low number of tools included in the distributions of such platforms limits the performance of a wide range of learning situations. On the other side, an increasing number of tools available on the Web, such as Google Docs[5], Twitter[6] or MediaWiki[7] are becoming very significant for educators. These are the main reasons that support the strong research concern about the integration of third-party external tools, in order to enhance the flexibility and ease of customization of VLEs and social networks [2][3]. In this line, well-known platforms like Facebook[8] or Google Wave[9] are explicitly relying on third-party efforts in order to extend the available applications, rather than developing their own tools in-house.

However, the integration problem when tackling with multiple tools and different platforms does not seem to find a general solution. Many political and technological issues can be considered in this regard. The main political issue is that the different vendors, understood as tool providers and VLE or social network providers, decided to impose their own *contracts* [4] to extend their features, making cumbersome to accomplish the integration of multiple external tools. Despite the efforts of different consortia to provide some standards that may allow the interoperability between these tools and platforms, such as IMS Learning Tools Interoperability Guidelines [5], the main vendors have not adopted them in real practice. Many technological aspects can be considered into these *contracts*, including the communication interfaces, the data exchange format, the distribution and access technologies, and even the security mechanisms involved. The heterogeneity between all these contracts normally entails a high *integration cost* to perform the integration of a tool into a platform. This cost is mostly related with a *software development effort* that must be assumed by those agents willing to complete such integration. This effort may vary regarding several design issues that have been already analysed in [4], e.g. the degree of coupling, the functional specificity, and the number of contracts offered by the different vendors.

[1] http://moodle.org
[2] http://lamsinternational.org
[3] http://sakaiproject.org
[4] http://elgg.org
[5] http://docs.google.com
[6] http://twitter.com
[7] http://mediawiki.org
[8] http://facebook.com
[9] http://wave.google.com

In this context, several authors and organisations have proposed their own approaches to decrease this integration cost and thus, the development effort that must be assumed to integrate multiple external tools in different VLEs and social networks. Some of the most representative approaches are: the aforementioned IMS Learning Tools Interoperability Guidelines (sometimes called Full LTI), a complex specification focused on the integration of web services; Basic LTI [6], a subset of the Full LTI specification that supports a lighter integration of web applications; the Apache Wookie (Incubating) [7], an architecture that enables the integration of W3C widgets and OpenSocial compliant tools in different platforms; and the GLUE! (Group Learning Uniform Environment) architecture, an evolution of the work presented in [8], allowing a loosely-coupled integration of tools developed with multiple technologies in VLEs.

The aim of this paper is to analyse the main features of each of these approaches in order to establish a comparison that may help to decide the adoption of some of them depending on the expected requirements. This analysis includes: the number and diversity of tools that can be integrated; the platforms in which these tools can be integrated; the degree of coupling involved; the richness in the communication and in the data exchanged between the platform and the tool; the opportunities for tool configuration; the possibility of using the same groups and roles that are defined in the learning platforms; their degree of standardization; the security issues involved; the development effort that must be assumed by the community of developers supporting each approach; and their current development status.

The remainder of this paper proceeds as follows. Section 2 introduces the four main approaches for the integration of external tools in learning platforms. Next, Section 3 discusses the main comparison aspects involving these approaches. Conclusions and future lines are presented in Section 4.

2 INTEGRATION APPROACHES

There are many approaches tackling the integration problem in the literature [4]. Nevertheless, this paper only considers those that try to design a global solution for different tools and learning platforms.

2.1 IMS Learning Tools Interoperability

IMS Learning Tools Interoperability, also known as Full LTI, is a specification defined by the IMS Global Learning Consortium in 2006 to allow the integration of external tools and contents in learning platforms [5]. Full LTI defines a complex contract, aimed to a tight integration of a wide diversity of tools, ranging from simple web applications to domain-specific learning environments and assessment tools. The Full LTI architecture is depicted in Fig. 1, and is composed by two main elements, a Tool Provider (TP) in the tool side, and a Tool Consumer (TC) in the learning platform side. The TC must include a proxy representing the associated real tool, and both TC and TP must implement a collection of services (sometimes called Tool Interoperability Runtime), to deploy, configure, and launch the tools. These services have been designed for a varied range of purposes, including single sign-on user authentication, and passing back outcomes from the tools to the learning platforms.

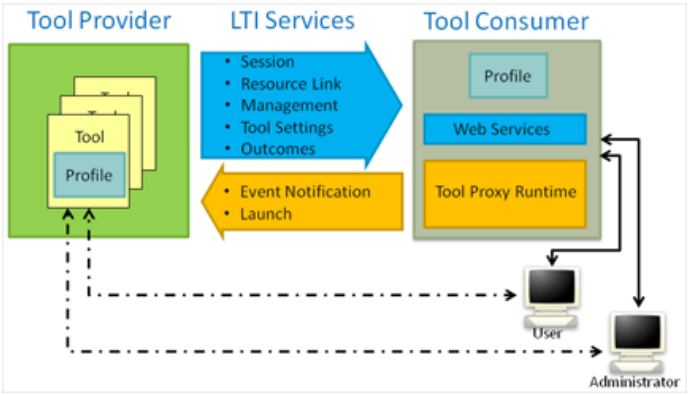

Fig. 1 Full LTI architecture. Figure taken from [9].

Full LTI needs an initial negotiation process between the TC and the TP where an agreement must be reached about the runtime services and the security policies that will be used, and about the set of addresses within the tool can be launched. Besides, Full LTI can be used in conjunction with the IMS Learning Information Services (LIS) specification [10], which defines the way to exchange information

about users, groups, courses and outcomes in a learning environment, and with the IMS Common Cartridge (CC) specification [11], which defines the format to distribute learning contents.

Nevertheless, the main vendors have been reluctant to develop the functionality of Tool Providers and Tool Consumers due to the complexity Full LTI. In fact, none of the current learning platforms is ready to support it. In 2010, the IMS Global Learning Consortium decided to define a new complex specification inspired by approaches such as the Building Blocks integration solution used by the Blackboard[10] commercial VLE. This specification is called IMS LTI v2.0 and is currently under development, only available for IMS members. At the moment, only Blackboard has announced support to this new version [12].

2.2 Basic LTI

Basic LTI [6] is a subset of the IMS Learning Tools Interoperability specification that has also been released by the IMS Global Learning Consortium in 2010. Basic LTI simplifies the Full LTI integration contract focussing on the loosely-coupled integration of web applications. Its architecture is depicted in Fig. 2, and shows an important reduction in the main elements, the Tool Provider and the Tool Consumer, and also in the number of additional services.

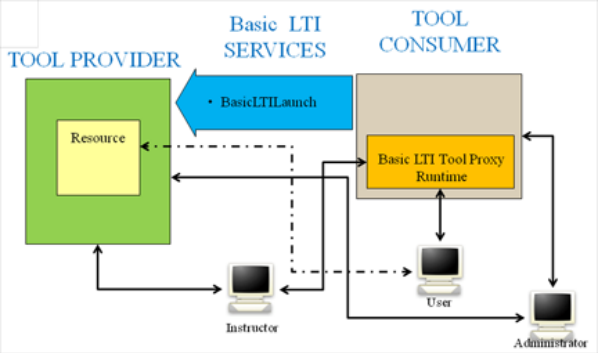

Fig. 2 Basic LTI architecture. Figure taken from [9].

Basic LTI only exposes a single address and a single POST method to access to the functionality of external tools. Therefore, the interactions between the platforms and the tools are very limited. Security issues have been drastically reduced, and single sign-on is no longer granted. The OAuth[11] protocol is used for encoding a shared secret sent by the TC, in order to authenticate it against the TP. This entails that the TC needs to be aware of the address URL as well as of the OAuth main parameters. On the other side, the TP must implement an OAuth compliant service to expose the URL address and to check the shared secret sent from the TC. Despite there are no mechanisms to pass back information to the learning platform, Basic LTI can optionally be used with some of the IMS LIS.

Vendors have reacted positively to Basic LTI and some of the main VLE providers, (e.g. Sakai or Desire2Learn[12]) are already implementing Basic LTI, while others such as Moodle or LAMS have this specification under development [12]. Additionally, Basic LTI is in IMS CC, thus it can take advantage of their publishers. This significant community supporting Basic LTI may mean an important success.

2.3 Apache Wookie (Incubating)

Apache Wookie (Incubating)[13] or the Wookie server is an implementation of the widget server architecture [7], a standalone server application that can manage W3C widgets [13]. This architecture, which is depicted in Fig. 3, provides a REST-based [14] contract that allows: the instantiation and the subsequent management of widget instances, the configuration of these instances, the installation and deployment of new widgets, and the storage of persistent information that can be shared between them. The Wookie server provides around sixteen native widgets (e.g. a chat, a forum, or a drawing tool) in its distribution, but it also supports most other W3C-compliant widgets, including thousands of them currently offered from Vodafone, Opera and Blackberry. Unlike the other integration approaches, the Wookie server has not been specifically designed for the integration of learning applications.

[10] http://blackboard.com
[11] http://oauth.net/core/1.0
[12] http://desire2learn.com
[13] http://incubator.apache.org/wookie

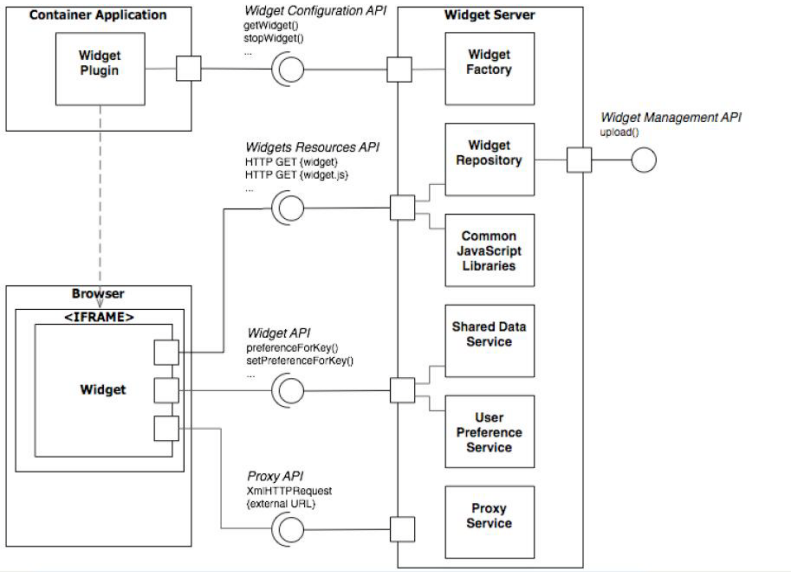

Fig. 3 Widget Server architecture. Figure taken from [7].

W3C widgets can be integrated in different learning platforms. To do so, a specific plug-in that shows the available widgets and asks for the creation and configuration of instances to a Wookie server must be specifically developed for each of these platforms. There are several plug-ins[14] that have already been developed for the Moodle and LAMS VLEs, for the Elgg social network, for the Wordpress[15] and Drupal[16] Content Management Systems (CMS) and for the JetSpeed 2 portal[17].

The Wookie server can also be extended with additional features that may be available depending on the requirements of each widget, such as the Bondi camera or the Wave Gadget API [15]. Besides, it can be combined with the Apache Shindig[18] in order to integrate OpenSocial[19] compliant tools. As well as its own REST API contract, Wookie can also be integrated using Basic LTI, acting as an IMS TP[20].

2.4 GLUE!

The Group Learning Uniform Environment (GLUE!) is a recent architecture that emerged as an evolution of the initial approach proposed in [8], with the aim of decreasing the average development effort that should be made to support the basic integration of several external tools developed with multiple technologies in different VLEs. This architecture defines a REST-based contract, follows a simple and loosely-coupled integration model, and uses the extension interfaces provided by tools and VLEs. The main ideas behind the design and development of GLUE! have not been published so far, because this is a work in progress that still needs to be validated in real learning situations. Nevertheless, it seems interesting to compare this alternative with the previous approaches in order to cover the different visions of the integration problem.

The GLUE! architecture is composed by three main elements as it is depicted in Fig. 4. The VLE adaptors deal with the specificity of each VLE contracts. However, due to the heterogeneity in these contracts, every VLE needs its own adaptor. They can be installed as any other VLE module or extension, providing an interface for the interaction of educators and students with GLUE!. Its main functionality is to show the list of available tools, and afterwards to request the creation and configuration of tools instances depending on the selected tool, but also on the number of users and groups that have been defined for that activity in the VLE. The GLUElet Manager stores a list with the available tools and the proper information to support their integration in an internal registry, and manages the process of creating and configuring instances. Finally, the Implementation Model Adaptors deal with the tool contracts. Several tools with the same technological implementations may use the same adaptors. These adaptors can create tool instances due to their knowledge about the tool contracts. They also contain specific or generic configuration templates for the tool they abstract.

[14] http://getwookie.org/Plugins.html
[15] http://wordpress.org
[16] http://drupal.org
[17] http://portals.apache.org/jetspeed-2
[18] http://shindig.apache.org
[19] http://code.google.com/intl/en-En/apis/opensocial
[20] http://code.google.com/p/basiclti4wookie

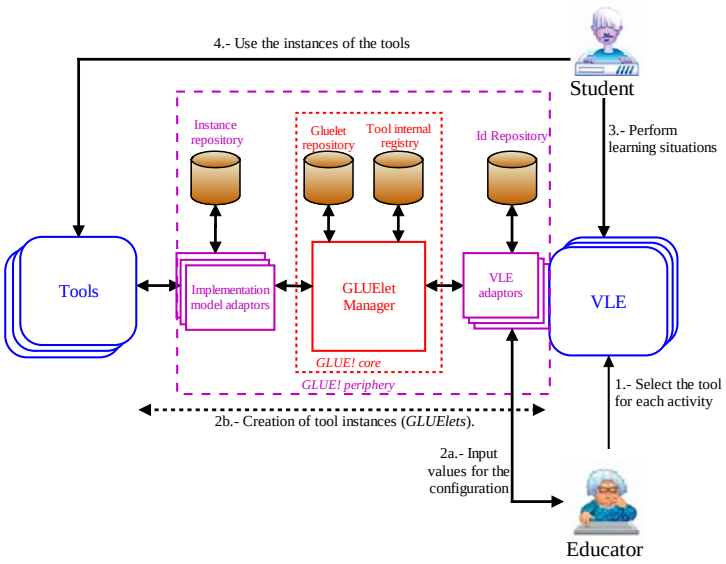

Fig. 4 GLUE! Architecture, including the three main elements: VLE adaptors, GLUElet Manager, and Implementation Model Adaptors.

The GLUE! architecture offers a light integration of the basic features of external tools, but keeping the main functionality of VLEs, regarding the definition of groups, roles and learning designs. This architecture relies on the development of the adaptors (GLUE! peripheral elements) by third-parties, in order to increase the number of tools that can be integrated in different VLEs.

3 ANALYSIS OF THE INTEGRATION APPROACHES

This section analyses the four integration approaches that have been introduced in Section 2 according to several important aspects, such as the number and diversity of tools and platforms that are supported; the degree of coupling and the interactions that are involved in the communication; the opportunities for tool configuration regarding the different groups and roles that are defined in the platforms; their degree of standardization; the security issues involved; and the development effort that must be assumed as well as the effort already done reflecting the current development status.

3.1 Number and diversity of tools

Full LTI and Basic LTI can theoretically integrate any external tool as long as there is a Tool Provider implemented for such tool. However, Full LTI is more oriented to a tight integration of “big” applications and SOAP web services [16] while Basic LTI is more oriented to a loosely-coupled integration of REST-based web applications [14]. Full LTI TPs need to implement a Full LTI compliant interface to receive requests from the Tool Consumers to the different addresses they exposed, and then perform different actions depending on each specific tool contract. In addition, they must support all the services defined in the Learning Tools Interoperability Guidelines to communicate with the external tool. Basic LTI TPs just need to implement a simple REST Basic LTI compliant interface to receive requests from the TCs to a single address, and then perform the required actions on the tools. Optionally, they can implement some of the LIS services to support additional functionality.

Similarly, GLUE! can theoretically integrate any external tool, as long as there is an Implementation Model Adaptor available for that tool. This approach is also oriented to a loosely-coupled integration of web applications, but can be extended to other technologies. Implementation Model Adaptors need to implement a REST-based interface to receive requests for the creation and management of instances for the tools they abstract, and the logic to transform such requests in order to be understood by the specific tool contracts.

A somewhat different approach is followed by the Wookie server because it can only integrate W3C widgets (and Open Social compliant tools when accompanied by the Apache Shinding). Tool vendors that do not follow the W3C contract cannot integrate their tools. The advantage is that no tool adaptors or plug-ins are required in the tool side to accomplish the integration of these widgets. Besides, the potential number of widgets is very large compared with the tools in the other integration approaches, though these will not principally be targeted at education.

A general requirement in the loosely-coupled integration approaches is that an URL representing the tool instances (GLUE! and Wookie server) or the TPs (Basic LTI) is used to embed the external tools (typically as an IFrame) in the learning platforms. Hence, applications that do not comply with this requirement must be wrapped by the corresponding adaptor (Implementation Model Adaptor or Tool Provider) in order to return these URLs to the VLE. The main problem is that most of the tracking VLE systems do not work with these light integration approaches.

3.2 Platforms in which the tools can be integrated

Full LTI and Basic LTI can be used in a different range of platforms (e.g. VLEs, CMSs, social networks, portals, etc.) as long as they implement their specific Tool Consumers. Once again, the Tool Consumer for Basic LTI is simpler than the one for Full LTI, due to number of services that must be implemented in the last one. Similarly, the Wookie server allows the integration of widgets in a wide set of learning platforms, as long as a specific plug-in has been developed to request the creation and management of tool instances.

On the other hand, GLUE! has been designed with a clear distinction in the functionality available for the main learning roles (educators and students). Only those playing the role of educators are allowed to create and manage tool instances, while both, students and educators, are allowed to see and modify their content. Thus, GLUE! is more oriented to VLEs, and cannot be used in platforms that do not allow the definition of such roles.

3.3 Degree of coupling involved

Full LTI promotes a tight integration of tools through SOAP-based interfaces. Moreover, it defines several additional services with the aim of monitoring, assessing and returning information to the learning platform. Nevertheless, a tight integration normally implies a higher cost, due to the high development effort that must be assumed to integrate several tools in multiple learning platforms [4].

In contrast, Basic LTI, the Wookie server and GLUE! define loosely-coupled integration architectures through REST-based interfaces, that enable the communication from the learning platforms to the external tools through adaptors that comply with their specifications. The main drawbacks are that none of them has been designed to provide useful data for interaction analysis from the participants in the learning situations, neither to pass information back to the platforms. In any case, additional APIs could be offered independently of the VLEs to enable a wider user base. This is applied, for instance, in the IMS Outcomes [10] LIS service, that can be optionally used in conjunction with Basic LTI, in order to return some relevant information, such as the grades of the students.

Finally, about the coupling between the integrated tools themselves, it is remarkable that the W3C widgets can share common data, which is stored in the Wookie server [17]. In Basic LTI and GLUE! the available external tools do not have any relationship among them, and so, they are completely independent.

3.4 Richness in the communication between the tool and the system

Full LTI may support a high richness in the communication and the data exchanged between learning platforms and tools. This is due to the specificity of TCs and TPs respectively that will normally entail their *ad hoc* implementation to comply with the specifications and the additional services. In contrast, Basic LTI presents a poorer communication with tools, because only one `POST` request is performed to launch the Basic LTI Tool Provider. Nevertheless, almost thirty data items might be included in such request, according to [6], so that the context (user, role and course) in which the external tool will be used is completely defined.

In between, the Wookie server supports a richer communication, compared with Basic LTI, in order to create instances and configure some of their properties afterwards. GLUE! also allows the configuration and creation of instances as well as their management, deleting them when the activity is finished. In addition, GLUE! shows to the educator in the VLE interface, a list with the available external tools that can be integrated, something that has not been considered in the other approaches. Nonetheless, GLUE! and the Wookie server are currently considering less number of data items compared with Basic LTI, despite including the main relevant information about users and roles. This is partially due to Basic LTI being a subset of Full LTI, and also related to LIS. This entails a lot of administrative information that is unlikely to be useful for the majority of more generic tools.

3.5 Opportunities for tool configuration

Full LTI considers an *ad hoc* Tool Setting service that can be used to modify the behaviour of the tools. This additional service must be implemented by TCs and TPs according to the potential configuration of tools. Basic LTI has different features including custom parameters that allow multiple variations of the tools that are exposed. These variations depend on the values that are introduced by administrators of learning platforms when configuring the TCs. However, these parameters are fixed for every TP and cannot be modified by the educators in the learning design process.

The Wookie server and GLUE! have defined different methods in their APIs to allow the configuration of the external tools by the educators. Their main difference is that GLUE! supports an *early binding*, defining the configuration of the tools before creating the instances, while the Wookie server supports a *late binding*, modifying the configuration of the tools after creating the instances. The Implementation Model Adaptors in GLUE! can contain different configuration templates that are rendered in the VLE interface for the educators when selecting the tools. Then, they can input different values (e.g. the title or an initial file in a Google Document) for each parameter. On the other side, the properties of the widget instances can be also modified through the interface shown in the learning platforms.

3.6 Possibilities of using the same groups and roles that are defined in the learning platforms

Basic LTI and Full LTI consider the concept of *addresses*, instead of the concept of *instances*, such as in GLUE! and in the Wookie server. In fact, Basic LTI considers a single address to launch each TP, and so, all the students will access to the same abstraction of a tool. To support several groups, different TPs should be implemented for each external tool. Besides, an administrator should manually configure the URL address for each TP in the learning platform. Finally, the educator should match each TP with each group. This approach entails several manual steps that increase the burden of both administrators and educators. In contrast, Full LTI considers different addresses for each Tool Provider, thus different users and groups can access to different abstractions of the same tool.

The Wookie server and GLUE! have been designed to create multiple instances of the external tools. Each instance represents a new client of the tool, configured and ready to be used by the final users. These instances can be assigned to one single user or to a group of users, and they are created by the educators when designing their learning activities. It is interesting to point out that the Wookie server and the Apache Shindig server allow the creation of sibling instances of a tool for different learning platforms at the same time (e.g. a chat widget shared between a group in Moodle and in Wordpress). Finally, the four approaches consider the use of roles in their specifications with different use cases for the main actors: administrators, educators and students.

3.7 Degree of standardization

Full LTI was released as a complex and detailed specification in 2006 with the name of IMS Tool Interoperability Guidelines [5]. It is based on some commonly-used protocols and standards (e.g. SOAP and HTTP for the service interfaces, or WS-Security for the signing and encryption in the communication). Nevertheless, this specification has not been adopted by the main vendors and, thus it has not become a standard. On the contrary, Basic LTI had a better response among vendors as an IMS internal draft, and afterwards, when the specification was released in the beginning of 2010. Hence, it is expected to become a standard soon. Basic LTI is also based in commonly-used protocols and in the REST architectural style for the communication between the elements.

The Wookie server is currently under incubation at the Apache Software Foundation in order to spread their code among users and developers. The widgets that can be integrated in the Wookie server must be developed and distributed according to the W3C specification [13]. The integration approach with the learning platforms follows a draft specification based on a REST API[21]. Besides, the Wookie server implements the functionality to be compatible with the Google Wave Gadget API and with the OpenSocial specification through Apache Shindig.

Finally, GLUE! propose a REST-based interface among their element that is currently being evaluated and redesigned by their authors, before publishing a first draft specification. The data is exchanged

[21] http://incubator.apache.org/wookie/wookie-rest-api.html

using the standardised Atom[22] syndication format, while the configuration templates that can be used by the educators to configure the different instances follow the XForms[23] specification.

3.8 Security issues involved

Full LTI supports multiple authentication and authorization models through the use of security profiles, as long as they are implemented in TPs and TCs. Profiles are sent in SOAP headers when launching the requests and receiving the outcomes. These security profiles are an extension of the WS-Security specification, which support a shared-secret approach for the authentication between the TC and the TP. After that, a session can be established for the communication between the learning platform and the tool. There are some concerns here for user privacy and safe harbour of data (e.g. Blackboard in a UK college sending student details to a publisher in the US, for storage outside of safe harbour).

Basic LTI uses the OAuth protocol in the requests from TCs in order to grant the access to TPs. Therefore, they both need to implement OAuth and share a common secret and key for each TC-TP combination. It is important to remark that the user is not involved in this process and so, single sign-on with the external tools is not granted; instead, the authentication problem when accessing third-party resources is moved to the implementation of TPs.

The Wookie server does not support any authentication mechanism to access the widgets functionality because they do not require it. Nonetheless, a stardardised OAuth solution coming from the Apache Amber[24] project is under research to be applied along with the Apache Shindig. Presently, widgets need to implement their own authorization mechanisms to communicate with external web APIs. The Wookie server does however implement a whitelist proxy service to prevent unauthorized data sharing between widgets and external APIs.

The GLUE! architecture is currently searching a compromise solution to provide single sign-on when accessing to external tool instances from the user's credentials in the VLE. In this context, different approaches have been studied. The first one utilises institutional credentials that are stored in the internal registry identifying each of the available tools; these credentials are transparent for the final users, and support the internal authentication between the Implementation Model Adaptors and the tools. A second approach uses an external server connected to the VLE, in which users should store their credentials in the external tools. The drawback in the first approach is that multiple users and institutions would be using the same credentials with the consequent security problems involved. The limitation in the second approach is that students and educators are forced to create different accounts in the tools; furthermore, all this information becomes centralised with the consequent security risks if the external server is hacked. An OAuth approach has been also considered in GLUE! with the drawback of making cumbersome the development of the adaptors and with the limitation that currently not every external tool implements this protocol.

3.9 Development effort that must be assumed

The four integration approaches need a development effort to add new tools and new platforms. This effort must be assumed by the integration agents when developing the adaptors, plug-ins, extensions, or modules, according to the different specifications and architectures.

The implementation of a Tool Consumer and a Tool Provider according to Full LTI is a time consuming and a complex task. This task entails the development of many additional services that are exposed in the specification, as well as the *ad hoc* implementation of the different SOAP APIs. In contrast, Basic LTI just needs to implement a few methods in the TC (in order to gather the different data items from the learning platform and create the POST request), and in the TP (in order to process the requests coming from the TC and communicate with the external tools). Optionally, different additional services can be implemented depending on the purpose of the integration and on the effort that can be devoted. The more additional services, the richer the integration, but also the more effort devoted. Nevertheless, if a new TC is implemented for a specific platform all the Tool Providers that were already available for other platforms become automatically useful for the new TC, according to the Basic LTI specification.

The same happens in GLUE! with the VLE Adaptors and the Implementation Model Adaptors. The external tools integrated in Moodle will become available for LAMS when the correspondent VLE

[22] http://tools.ietf.org/html/rfc4287
[23] http://w3.org/TR/xforms11
[24] http://incubator.apache.org/projects/amber.html

adaptor is developed. In this approach, the effort in the tool side depends on the tool contracts; web applications can be easily integrated, while others need to wrap their functionality to allow the creation, configuration and management of tool instances. The VLE side is a little more complicated because the VLE has to be extended, so that it can offer educators the list of available tools and the configuration templates, and also embed the URLs from the tools instances in the VLE interface.

The Wookie server adaptors for the learning platforms just need to implement a REST API that creates and configures the instances and embeds the widgets in the platform. For this approach, most of the development effort falls in the implementation of the functionality of existent external tools, developed with multiple technologies, as W3C widgets.

3.10 Development status

Basic LTI is the approach with the higher development status in the middle of 2010, due to the important community behind this specification. Several TCs have already been implemented for the main learning platforms [12], such as Sakai, WebCT or Desire2Learn, while the ones for Moodle and LAMS are under development. Additionally, some TPs are also available, including those for MediaWiki and the Wookie server. In contrast, Full LTI has not succeeded, and thus, just a few TPs have been developed as a proof of concept, while none of the main vendors has make an effort to comply with this specification. The complexity of the new IMS LTI v2.0 might entail the same problem.

The Wookie server can be instantiated and integrated in different platforms that have been mentioned before, such as Moodle or LAMS, and there is also a Connector Framework for writing new plug-ins in PHP, Java, C#, Python and Flex. Moreover, there are several works trying to implement third-party tools as W3C widgets [18]. Therefore, it is expected that a large number of W3C widgets will be available in short time in the Apache Wookie (Incubating) distribution.

Finally, there is a prototype of the GLUE! architecture that includes a Moodle Adaptor, and multiple Implementation Model Adaptors for Google Docs, the Dabbleboard[25] shared whiteboard, MediaWiki and the Wookie server. This prototype supports the configuration of such tools using the same groups that are defined in the VLE, and accessing to the content of these tools through institutional credentials. There is another LAMS adaptor that is being developed for GLUE!, and more Implementation Model Adaptors are already scheduled. Nevertheless, GLUE! is still a work in progress, and there is not a community of developers to speed up its development status.

4 CONCLUSIONS

This paper has analysed the four main approaches that try to integrate multiple external tools in different learning platforms in order to increase the range of learning situations that can be performed on such platforms. This analysis covers up to ten important features of these approaches and compares the alternatives that have been followed when designing and developing their architectures and specifications.

The prospects suggest that loosely-coupled integration approaches have a good chance to succeed due to the reduction of the integration cost, and so the development effort that must be assumed to accomplish the integration, and because of the increasingly large number of web applications that can be easily and freely accessible on the Web. Therefore, complex approaches such as Full LTI are normally pushed into the background, used only for specific integrations, and replaced by lighter specifications such as Basic LTI.

Nonetheless, lighter specifications present important limitations because of their generic interfaces and contracts. These limitations may affect the communication between learning platforms and tools, but also the discovery of the available external tools, the opportunities for tool configurations, and the technologies and security mechanisms that are supported. Basic LTI, the Wookie server and GLUE! have been thought to deal with these features in many different ways, sometimes stressing their design to consider one of these features, and sometimes not.

The important backing of a consortium as IMS GLC, and its community of supporters will probably make Basic LTI the reference standard for the integration of external tools in real practice. However, the recent trend to implement the functionality of different applications as W3C widgets may enhance the spreading of the Wookie server. In contrast, the research work in GLUE! has a long way ahead.

[25] http://dabbleboard.com

ACKNOWLEDGEMENTS: This work has been partially funded by the Spanish Ministry of Science and Innovation (TIN2008-03023) and the Autonomous Government of Castilla y Leon, Spain (VA106A08). The research stay in which this work was done was funded by the Spanish Ministry of Education (EDU/2933/2009).

REFERENCES

[1] C. R. Graham, C. R. Blended learning systems: Definition, current trends and future directions. In C. J. Bonk & C. R. Graham (Eds.), The handbook of blended learning: Global perspectives, local designs, pp. 3-21. San Francisco, CA: Pfeiffer Publishing. 2005.

[2] C. Severance, J. Hardin, and A. Whyte. The comming functionality mash-up Personal Learning Environments. *Interactive Learning Environments*, 16(1):47–62, 2008.

[3] J. Fontenla, M. Caeiro, and M. Llamas. Towards a Generalized Architecture for the Integration of Tools in LMSs. *International Journal of Emerging Technologies in Learning (iJET)*, 4:6–11, 2009.

[4] C. Alario-Hoyos, J.I. Asensio-Pérez, M.L. Bote-Lorenzo, E. Gómez-Sánchez, G. Vega-Gorgojo, A. Ruiz-Calleja. Integration of external tools in Virtual Learning Environments: main design issues and alternatives. *Proceedings of the 10th International Conference on Advanced Learning Technologies*, ICALT, pp. 384-388, Sousse, Tunisia, July 2010.

[5] IMS Tool Interoperability Guidelines. Version 1.0. URL: http://www.imsglobal.org/ti/tiv1p0/imsti_guidev1p0.html. Last visited: September 2010.

[6] IMS GLC Learning Tools Interoperability Basic LTI Implementation Guide. Version 1.0. URL: http://www.imsglobal.org/lti/blti/bltiv1p0/ltiBLTIimgv1p0.html. Last visited: September 2010.

[7] S. Wilson, P. Sharples, and D. Griffiths. Distributing education services to personal and institutional systems using Widgets. In *First International Workshop on Mashup Personal Learning Environments*, pp. 25–32, The Netherlands, September 2008.

[8] J.I. Asensio-Pérez, M.L. Bote-Lorenzo, G. Vega-Gorgojo, Y. Dimitriadis, E. Gómez-Sánchez, and E.D. Villasclaras-Fernández. Adding mash-up based tailorability to VLEs for scripted Collaborative Learning. In *First International Workshop on Mashup Personal Learning Environments*, pp. 14–17. Maastricht, The Netherlands, September 2008.

[9] IMS GLC: Learning Consortium: Learning Tools Interoperability v1.0 Project Group. http://www.imsglobal.org/toolsinteroperability2.cfm. Last visited: September 2010.

[10] IMS GLC Learning Information Services Specification Primer. Version 2.0. Public Draft Release Version 1.0. http://www.imsglobal.org/lis/lisv2p0pd/LISspecPrimerv2p0pd.html. Last visited: September 2010.

[11] IMS Common Cartridge Profile. Version 1.0 Final Specification. URL: http://www.imsglobal.org/cc/ccv1p0/imscc_profilev1p0.html. Last visited: September 2010.

[12] Common Cartridge and Basic Learning Tools Interoperability Progress. URL: http://imsglobal.org/cc/statuschart.html. Last visited: September 2010.

[13] Widget Packaging and Configuration. W3C Candidate Recommendation 01 December 2009. URL: http://www.w3.org/TR/widgets/. Last visited: September 2010.

[14] L. Richardson and S. Ruby. RESTful Web Services. O'Reilly Media, Inc., May 2007.

[15] S. Wilson, P. Sharples, D. Griffiths, and K. Popat. Moodle Wave: Reinventing the VLE using Widget Technologies. In *Second International Workshop on Mashup Personal Learning Environments*, pp. 47–58. Nice, France, September 2009.

[16] C. Pautasso, O. Zimmermann, and F. Leymann. RESTful Web services vs. "big" Web services: making the right architectural decision. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pp. 805–814, Beijing, China, April 2008.

[17] T. Nelkner. An Infrastructure for Intercommunication between Widgets in Personal Learning Environments. In *2nd World Summit on the Knowledge Society WSKS 2009,* pp. 41-48, Crete, Greece, September 2009.

[18] M.Á. Conde, F.J. García, M.J. Casany and M. Alier. Open Integrated Personal Learning Environment. Towards a new conception of the ICT-based learning processes. In *Proceedings of the 3rd World Summit on the Knowledge Society, WSKS 2010, (accepted for publication).* Corfu, Greece, September 2010.