Mashing up the Learning Environment

Evaluating a widget-based approach to Personal Learning Environments

Fredrik Paulsson
Interactive Media and Learning (TUV/IML)
Umeå University
Umeå, Sweden
fredrik.paulsson@edusci.umu.se

Abstract — **Different types of Virtual Learning Environments (VLE) have evolved and there is a steady ongoing progression of different concepts. During the last 10-15 years Learning Management Systems have dominated. Learning Management Systems are often presented as the solution for a range of educational needs. This paper presents a study of a mashup approach to the VLE using web widgets. A prototype was developed and discussed, covering technological aspects such as modularity, integration and adaptability as well as some pedagogical aspects, such as pedagogical flexibility and technological responsiveness. An alternative modular approach to the implementation of VLEs is suggested based on recent developments within web technology, stressing the use of standards and simplicity in order to address common problems of complexity and inflexibility resulting in poor conformance to pedagogical requirements.**

Keywords – LMS; MUPPLE; web widgets; mashup; VLE; PLE; e-learning.

I. INTRODUCTION

Different types of *Virtual Learning Environments (VLE)* have evolved over the years and there is a steady ongoing change and progression of different ideas and concepts for the VLE. During the last 10-15 years much has revolved around concepts Learning Platforms, such as *Learning Management Systems* (LMS). These systems are often presented as a common solution for a range of educational needs – much like a "Business System" for learning and education. However, the LMS have been criticized for being too inflexible and hard to adapt to different pedagogical contexts and needs (see, e.g., [1], [2] and [3]). The LMS are also criticized for having too much focus on the administrative aspects of learning with little support for pedagogical activities and pedagogical processes. Hence, having a strong focus on *Learning Management* rather than on actual learning and pedagogical activities per se - as the name actually suggests. From a system perspective LMS are commonly criticized for being designed and implemented in a silo-like fashion, contributing to lock-in effects of information and processes - very similar to the critique that is often heard about business systems in general. There is also a built-in conflict between the development and implementation of systems like LMS on the one hand and the development of social software and Web 2.0 on the other hand. While many LMS that are currently in use try to create a well-defined kind of "shielded community" for learning, web 2.0 is associated with open communities, global social interaction and open information services that can be used as building blocks for new services - such as for a *Personal Learning Environment* (PLE). However, observe that the notion of services for Web 2.0 refer to services that targets users and are not equivalent to services as in Service Oriented Architectures (SOA), which is to be regarded as a software design paradigm [4]. While the technology platform underlying Web 2.0 services may very well be a SOA platform, there is an unfortunate mix-up of those two rather different notions of services when discussing Web 2.0.

In order for services to be used as building blocks in such compositions (i.e., a mashup) the building blocks need to be well defined and with well-defined interfaces. Many web 2.0 services use proprietary interfaces such as the Twitter API, the Facebook API or APIs from Google and/or they use lightweight interfaces and protocols, such as RSS or Atom. This works well in many cases, but in order to build more sophisticated services and service compositions there is a need for more sophisticated interfaces and concepts for interaction [1]. This can obviously be accomplished by using advanced proprietary APIs, as illustrated in [5], but from a wider perspective, common open standards are preferable. This is also one of the issues the study discussed in this paper is set out to examine. The next section describes the state of the art, followed by a brief discussion of some central concepts and ideas related to some previous work, followed by a description of the presented study and the experimental implementation of a Mashed-up PLE. Finally the results of the study are discussed in the light of the ongoing progress and previous research in the field.

A. State of the art

While LMS-like system are typically implemented by most educational institutions, the movement within the teaching community as well in the research community is towards adaptive and responsive learning environments, similar to PLEs, see e.g., [3][6][7][15][28]. However, while pedagogical concepts like responsive learning environments are attractive, the technology currently in use doesn't support it very well. At the same time, education needs specialized services for dealing with pedagogical requirements, such as Personal Development Plans (PDP), digital portfolio, services for discovery and integration of digital learning resources, and so forth, which are resulting in several good and useful tools for learning, but they are not well integrated with the rest of the VLE [1][19][28]. These and similar issues are often addressed through different approaches to system integration, such as using proprietary APIs or more general integration by Web Service technology [1][4][5][14].

However, such approaches to building the Learning Infrastructure has turned out to be problematic for several reasons. Firstly, it becomes expensive to integrate “per system”, using proprietary APIs. API integration also makes the systems hard coupled, which supresses flexibility [1][10]. Secondly, using (commonly SOAP-based) Web Service technology tend to become very complex as well as expensive, adding an cost, as well as technical, overhead [1][10][13], which is also illustrated in the VWE case discussed in section B. And thirdly, by mixing a monolithic concept, like the LMS with a modular service based approach some of the technical flexibility needed for dealing with some of the pedagogical requirements is lost [1][3][6]. In recent years there has been a general development on the Internet towards modularity and an alternative kind of loosely couple services driven by less complex and more web friendly service integration, such as using RESTFul APIs [21] and lightweight APIs and protocols, such RSS and Atom combined with widget and mashup technologies [16][17][20], which are described in detail in section C. This development stands out as exceedingly suitable for the next generation of learning environments, fulfilling the flexibility requirements for personal and responsive learning environments by providing a standardized framework for modularity and loose integration on the web that is now being studies by the research community in general and in an education context [25][28][30][31].

B. The Personal Learning Environment

Simply put, a *PLE* can be described as a learning environment where the learner is in focus as well as in control of the learning environment. However, the main objective of the PLE is to put the learner in control of his own learning rather than in control of the learning environment, even though these two are obviously related. Learning is regarded as a constant, ongoing process, as is the evolvement and change of the learning environment. The learning environment needs to be responsive and adapted to different contexts, needs and pedagogical requirements. These are qualities that are commonly emphasized, such as in [1], [3], [6] and [7], to give just a few examples. One of the ideas that are often emphasized in relation to PLEs is that personal “tools”, such as blogs, twitter, etc., that are personal and used in other contexts can also be used as components of the PLE.

1) The Virtual Workspace Environment

The concept of a PLE is very similar (if not identical) to the idea underlying the *Virtual Workspace Environment* (VWE) that was first outlined in 1998, described in [2], even though the means to accomplish it were different. Simply put the VWE can be described as a component based VLE where users (i.e., teachers and students) can construct personal or shared learning spaces using a web browser.

In recent studies [1][5], it was shown that using modular approaches for the design and implementation of learning environments can address some of the LMS related issues, that were described in the previous section. A common modular taxonomy (*The VWE Learning Object Taxonomy*) for use with both VLEs and Digital Learning Resources (DLR) was presented in [2]. The taxonomy was compatible with the widely referenced *Learning Object Taxonomy* by Wiley [8] and demonstrated how the VLE and DLR could be implemented using a common modular, conceptual and architectural model that allowed for a common composition of both the VLE and learning content. Altogether this work resulted in two prototypes for composing and assembling modular VLEs; called the *Virtual Workspace Environment (VWE)*. The VWE was presented in [9], where the two different implementation approaches were compared. One using a JAVA RMI based approach and the other using a Web Service (SOAP) based approach. Both prototypes made it possible for teachers and/or learners to compose shared or personal learning environments by picking and choosing from a set of functional (software) components (called VWE tools). The VWE tools acted as building blocks providing the functionality for the learning environment. The ideas underlying the VWE were to a great extent inspired by the development of component-based software, as well as the fundamentals of Service Oriented Architectures (SOA), described in, e.g., [4][10][11][12].

A “proof of concept” was established, and by developing the prototypes using two different implementation approaches it was possible to isolate a couple of issues resulting from the taxonomy versus the model and the implementation approaches [9]. One of the problems that were identified was that, even though the use of standards was extensive (such as standards for Web Services, communication protocols etc.), the prototypes (and thereby the modular approach) only worked within the isolated context of the prototype environments and could not be generalized without new standards. This problem was mainly caused by a lack of standards supporting modularity for the creation of *Rich Internet Applications (RIA)* (see, e.g., [13]). In recent years, things have changed and standards have evolved and matured. Among the most interesting directions, from a modularity and RIA perspective, is the idea of Web Widgets and Mashups, see, e.g., [14][15][16][17]. The study presented in this paper starts out from the hypothesis that widget technology and widget mashups have the potential of overcoming many of the problems encountered during the VWE project [9], while still providing full support for the underlying ideas of modularity and the shift of central functionality and software from the desktop to the web, allowing for collaboration and social interaction with typical desktop functionality in ways that are only possible on the web. Furthermore, the creation of mashup learning environments can be adapted to different pedagogical scenarios and approaches in a dynamic and transparent way. Such transfer of functionality with its built-in potential has already been proven by services like Google Apps and other similar (web/cloud) services, see, e.g., [18] and [19]. However, the kind of rich functionality that is provided by such services needs to be put into context as an integrated part of the learning environment. Mashup Learning Environments (MUPPLE) are a step in this direction and it is also where the study presented in this paper and the WiMUPPLE project come in to play.

In retrospect, it can be said that the PLE concept is more Web 2.0 friendly and as such more flexible in terms of interpretation and implementation - with reference to choice and use (as well as “misuse”) of technology, whereas the VWE concept provides a more explicit architecture model for the technology platform in relation to modularity and composition of mashup environments [9]. However, those features have also made VWE proprietary as only components that follow the VWE conceptual model and architecture can be used as building blocks. As a result, the VWE has also become too complex and dependent on VWE services and APIs as shown in Figure 1.

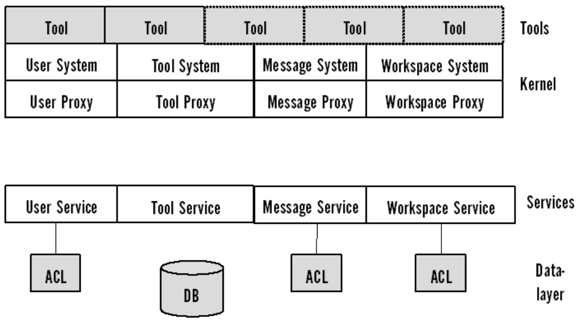

Figure 1. The figure shows an overview of the VWE architecture, the VWE Kernel and the VWE services used by tools to interact with the workspace.

In order for a component to work in the context of a VWE workspace, it needed to implement the VWE Service APIs, and all interactions with the workspace and other tools were via those server-side services. These were dependencies that severely limited the flexibility and usefulness of VWE from a Web 2.0 point of view.

For those reasons, one of the objectives of the WiMUPPLE project is to illustrate a third implementation strategy that addresses those problems and that makes the learning environment more generic, which is likely to be a characteristic needed in order for the concept of MUPPLEs to gain wider acceptance.

C. *Mashups and Widgets*

There are several (but similar) definitions of a mashup. A mashup is commonly defined as being a combination of different services on the web in a way that create a new composite application (or service) with added value. A widget-based mashup obviously uses widget technology and is currently typically constructed using a mashup environment such as Netwibes, iGoogle or our WiMUPPLE-environment [20]. A mashup can also be created by very simple means, using simple web tools that allow users to combine services on the web by matching and mixing information using lightweight interfaces such as RSS or Atom. However, in such cases it is mainly about mashing up information and not about mashing up functionality and services in a way that goes beyond the delivery and consumption of information. However, information mashups can be valuable in many cases, as part of a PLE.

Even so, if you are a developer or an experienced user you might want to use one of the more sophisticated approaches that are available for the development of web-based applications, or RIA as it is sometimes referred to.

The widget landscape is somewhat complex and can be roughly divided into three main categories: widgets for cellular phones (such as widgets for Android phones), desktop widgets (such as the widgets in OS X or gadgets in Windows) and finally web widgets, which are basically widgets that are distributed in the web browser [20]. *The widgets referred to in this paper are solely web widgets*. Even though these are three rather distinct categories there are several important similarities. One of the most significant similarities is that their implementations are based on web technology (or at least technologies that are commonly used on the web), such as html, JavaScript and XML (and AJAX). This is also an important property for sharing and reusing information and functionality since web technology relies on well-established standards. Besides the commonly used web standards there are widget-specific standards as well. However, widget standards are still rather untested and/or under development and there is still some way to go before it is possible to say that there are well established standards for widgets in the same sense as for the web. This means that there is always a trade off between the use of standards and proprietary widget technology when developing widgets that need sophisticated functionality.

The remainder of this paper presents and discusses a study that illustrates how widget technology can be applied to a modular concept, like the one previously described [2] and how a modular and web based learning environment can be implemented and assembled "on the fly" by learners and/or teachers. Both PLEs and LMS-like learning environments can be constructed in similar ways depending on the type of widgets, and supporting backend systems that are available. The same underlying SOA based server-side architecture that was used in the VWE project could in fact be used to support a client implementation using widgets, even though a REST based architectural model is preferred in order to avoid some of the complexity and limitations of the previous prototypes that were discussed above and in [9][21]. A RESTful approach also contributes to making integration with third

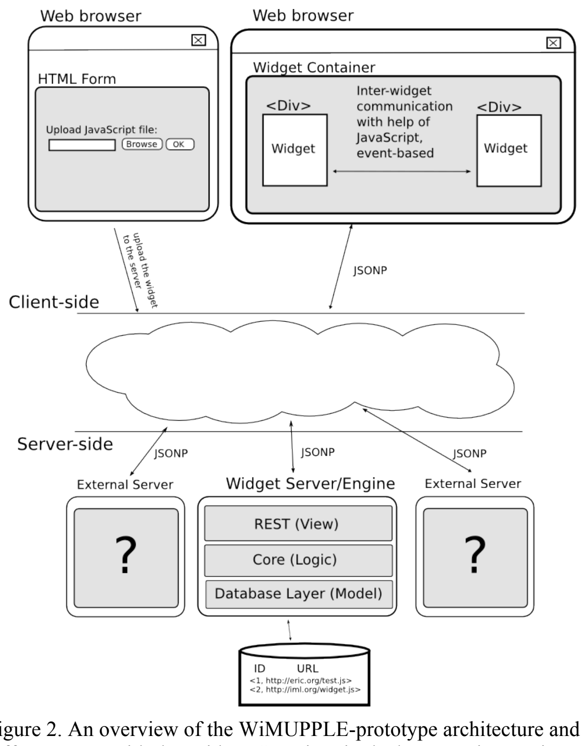

Figure 2. An overview of the WiMUPPLE-prototype architecture and its different parts with the Widget Container in the browser, interacting with the server layer via the REST API using JSONP.

party services easier and generally less complex and is more "web friendly", even though not all problems can be solved in a RESTful way.

II. Objectives

The main objective of the research presented in this paper is to explore how widget-based mashups can be used as a basis for constructing a PLE or *Mashup Personal Learning Environments* (MUPPLE) as PLEs are referred to when implemented as mashups. The mashup approach can be compared to the two approaches used in the VWE project that was briefly described above.

In addition, the widget based MUPPLE approach is applied to a similar modular concept that was presented by Paulsson and Berglund in [9], where it was illustrated how the modular concept of Learning Objects can be extended to also become a modular concept for the whole VLE (i.e., a PLE or an LMS) by adding some basic software architectural rules and principles that conjures a number of essential properties to the otherwise content centred concept of Learning Objects, see [22][2][1].

Another objective is to illustrate that the concept of a modular framework, such as the VWE Learning Object Taxonomy, can be applied using more Web 2.0 friendly and generic approaches. Therefore the work presented in this paper will be discussed and compared to the work presented in [2], where the VWE Learning Object Taxonomy was introduced and in [1][9], where the two VWE prototypes were discussed (also discussed above) in relation to pedagogical requirements and learning theories.

III. Methodology

Besides surveying the literature in the field, this study is based on an experimental approach where a prototype was developed and tested. It should be emphasized that even though this study addresses issues and requirements that emanate from a pedagogical standpoint, i.e., creating conditions for pedagogical adaptability and responsiveness in technology, the objective is not to evaluate the pedagogical implications at this point. The purpose is instead, which is also discussed above, to evaluate how the concept of a more generic and web friendly approach, using widgets and mashups, can be utilized from a technological standpoint to build MUPPLE, in comparison to earlier less generic implementations, such as the VWE. And furthermore - how to do this by applying existing concepts. However, in the discussion section of the paper the results are also discussed in relation to some pedagogical issues and implications based on experience from other studies, in order to better illustrate how modularity, technology implementations and pedagogical issues are linked.

1) *Technology settings*

An important starting point was to avoid developing everything from scratch. There are a multitude of ongoing development and project addressing widgets and mashups and whenever it has been possible existing work has been used.

The prototype architecture follows common design paradigms and patterns, illustrated by Figure 2, which also illustrates how widgets are handled on the client using a widget container that renders the widgets. The inner working of the widget container is illustrated in Figure 3 and described in more detail below. Even though Figure 2 illustrates a schematic architecture using a web browser as the client, the client could in fact be any other widget platform, such as a handheld device or dashboard widget. The widgets used for the purpose of the prototype are described from the point of view of being used in the context of a VLE, but in theory most of the widgets could be used in other contexts as well as they are often generic functional components that have been contextualized by the mashup and the pedagogical context.

JavaScript Object Notation (JSON) [23] is used for communication and data interchange between widgets and servers. As illustrated by Figure 2 and Figure 3, widgets that run on the WiMUPPLE platform can interact with a widget server, a widget engine or any other external server using JASON (or JSONP for managing cross domain interaction). This creates a flexibility that goes beyond the "local" ICT infrastructure and makes it possible for widgets to potentially interact with any servers that are of interest, acting as lightweight clients for other systems that may be relevant in the context of a learning environment. This differs from the previous VWE implementations in that it provides a transparent and generic infrastructure rather than a proprietary and platform dependent API.

This creates flexibility in terms of making the learning environment adaptable to different and chancing requirements. Furthermore, it makes the learning environment independent of a specific LMS vendor to implement certain functionality. It has proven to be quite straightforward to develop simple widgets that can act as clients to different legacy (as well as to other) systems.

Flexibility, in terms of being adaptable and distributed, is an essential property of a modular environment since it allows for the learning environment to be distributed (service-wise) over the Internet and at the same time it makes it possible to personalize and adapt the learning environment at the service level for group preferences as well as for personal preferences. It also creates the characteristics needed for responsive VLEs. These are important differences compared to the concept of an LMS, which has a centralized approach with clear system borders limiting the ability to interact with the surrounding world to the interactions that are countered by the LMS vendor or (in some cases) plug-ins and suchlike developed by third-parties, This also means that the functionality is limited to what is supported by the LMS, while functionality can be added and removed dynamically in the mashup PLE.

B. *The Widget Container*

The client hosts the widgets within the *widget container* (see Figure 2), which is loaded into the browser and rendered. Each widget has the possibility to communicate and interact with external servers as well as "internal" widget specific servers that are specifically developed to serve the widget. The widget container can actually be compared to the "kernel" in the VWE implementation. However, the kernel was implemented as a Java Applet, while the widget container relies on the JavaScript capabilities of the web browser and the standards associated with widgets and is therefore a more generic solution. Figure 1 illustrates the VWE kernel implementation, while Figure 2 illustrates the role of the Widget Container.

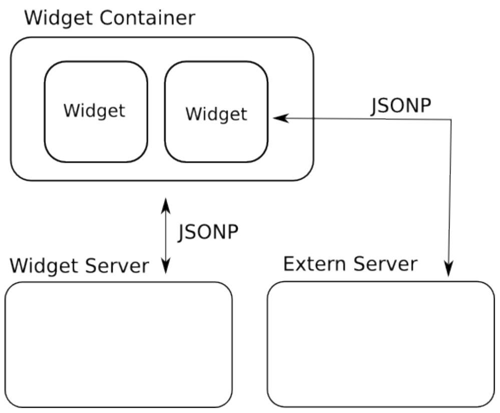

Figure 3. Illustrates the design and inner works of the Widget Container and how widgets interact with the widget server and/or external servers using JSONP.

Technically, the widgets used by the system consist of JavaScript that is loaded into the widget container where they are rendered and executed. The Widget Server keeps track of what widgets are available and the Widget Container communicates with the Widget Server using JSONP and the predefined RESTful API. Thanks to the Widget Container, it is possible to move the widgets around in the browser's workspace. Hence, the Widget Container also serves as the "glue" that holds the browser representation of the learning environment together and creates the feeling of an integrated environment in the same sense as the LMS. There is however an essential difference in the philosophy and approach underlying the integration. While the LMS relies on a strong, silo-like strategy for integration, the MUPPLE relies on loose integration of freestanding services and components.

C. *The Widget Server and the Widgets*

As previously mentioned, widgets are basically JavaScript uploaded to the Widget Container via the Widget Server. Besides the communication with the Widget Server, widgets can communicate with other external servers using JSONP. In the case of WiMUPPLE a choice was made to use *Yahoo Querying Language* (YQL) [24] for the implementation of the widget server in order to avoid unnecessary in-house development. However, it is fully possible to use other approaches as well, such as Google or other servers that are widget

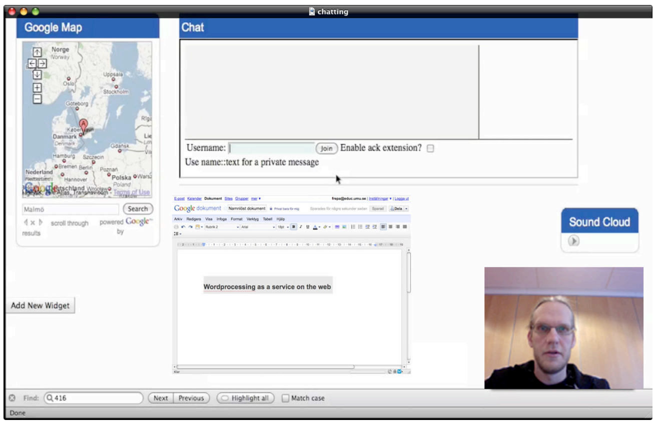

Figure 4. Screenshot of a learning environment created using WiMUPPLE with a number of widgets for different purposes.

specific, with similar results. This is actually not a big issue and is illustrated by Figure 2. Besides YQL, the WiMUPPLE Widget Server was built using the Python-based Django framework and a traditional MVC pattern.

IV. RESULTS AND DISCUSSION

The experimental implementation and the resulting prototype show that it is quite possible to implement a modular VLE using widgets and mashup technology. Or in the WiMUPPLE case, a framework for composing and administrating mashup learning environments in a way that can be managed by teachers and students (shown by Figure 4) and in which functionality can easily be added and removed. With the right set of widgets, a complete LMS could theoretically be built using the WiMUPPLE, even though an LMS is probably not what is wanted or needed.

The WiMUPPLE implementation makes it quite clear that it is less complex using a widget approach compared to the Java RMI and/or SOAP approaches used in the VWE-project [5]. However, the widget mashup approach is in general less powerful in terms of building sophisticated functionality. One of the main issues in this respect is the internal communication, i.e., inter-widgets communication. In an LMS everything is closely integrated, which is also what causes the main problem with the LMS concept, but at the same time it is a strength in terms of inter-system communication. All parts within the system can easily be made aware of all other parts in the system. In a mashup everything is loosely coupled and different widgets are normally completely self-standing and self-contained and not "aware" of the context in which they are used. This makes it harder to maintain the feeling of a well-integrated learning environment. The VWE implementation had similar problems that were solved by implementing a "Message Service" (see Figure 1) that managed the interaction between components and different parts of the learning environment, including other components (tools). The drawback with this solution, besides being proprietary, was that all the components became dependent on a common server side infrastructure in order to function in the context of an integrated learning environment. When working with widget-based mashups such solutions become a problem, as we want to be able to use any kind of widgets that follow the widget standards, i.e., not depending on a common server side infrastructure. An alternative solution would therefore be to make the widgets aware of each other within the web browser and allow widgets to interact and communicate with each other directly. This is technically possible and Sire et al. have described an example of such interaction in [25], where they discuss the implementation of drag and drop between widgets in the browser. There is currently no standardized or obvious way of implementing direct widget interaction and it will demand some tweaking to work. However, this is one of the issues that are likely to be solved by html5.

There has been some tweaking in order to get everything to work as expected, which was mainly caused by the immature nature of the widget technology concept compared to the maturity of Java RMI and SOAP. However, it is highly likely that the adoption of html5 will solve many of the problems and issues encountered here as well.

The fact that the principle of modular VLEs is realistic was shown in [1][9] it was discussed how modularity contributes to creating important pedagogical advantages related to flexibility and adaptability, which are qualities needed to create learning environments that are responsive and adaptive to users needs and to changing pedagogical requirements. The WiMUPPLE add to those characteristics by using generic and web friendly technology that open up for a much wider range of components to choose from.

Furthermore, modular learning environments are better adapted to suit different learning theories and pedagogical approaches as well as to changing pedagogical scenarios. Such features are beneficial, even essential, in many learning scenarios, especially when working with pedagogical methods and approaches like Learning, where it is hard (if not impossible) to foresee the learning path from start to finish beforehand – and thereby also to foresee the needs of the learning environment. These are also the main reasons why it is important to continue the research and development of modular concepts for VLEs - like the WiMuPPLE. Taken as a whole, the project also illustrates potential business cases where market competition is opened up for smaller actors to compete with LMS vendors by providing small and specialized components acting as building blocks in a mashup learning environment.

It has already been shown that modular learning environments hold an interesting pedagogical potential. In [1], it was illustrated that there is a correlation between modular environments and adaptability and responsiveness and that such features create pedagogical flexibility. The experimental study presented in this article shows that, not only is it possible to build modular learning environments, but it can be done using web based standard technology that bears the potential of almost endless flexibility in terms of access to functional components – in this case widgets. In the long run, this means that generic components (i.e., widgets) can be integrated as a part of a modular VLE without the need of adding learning specific code or support for certain APIs, even though such APIs may be beneficial in many cases, something that is discussed below. Even though the experiments showed that this can be technically accomplished (even if the technology is still somewhat immature) there is still a need for a better "glue" to tie mashup learning environments together and to create pedagogical context.

A. 3.2 Future work and developments

There are some very intriguing progresses around the corner that are likely to benefit the development of mashup learning environments. On the one hand there is the general development, such as the gradual evolvement of html5 and standards for widgets and mashups. On the other hand there are developments within the field of learning technology standards that, at least on paper, look very promising from a modular learning environment perspective. Among the most interesting developments are the new specifications from IMS: IMS Common Cartridge (IMS CC) [26] and especially the IMS Learning Tools Interoperability (IMS LTI) [27]. We are currently in the process of examining whether IMS CC can be used as a packaging format for our widget-based MUPPLE and furthermore, if IMS LTI can be used as a standard for widget communication and interactions within a widget-based VLE. Severance et al. have already described some experiments in [28] where IMS LTI was tested in a mashup environment and the results seem promising and could be taken even further in the WiMUPPLE environment.

Another direction, that has already started, is the integration of the *Spider* and the WiMUPPLE environment. The Spider is a national search service for digital learning resources that connects a number of repositories, using either metadata harvesting or search federation, in a way that makes it possible to search for learning resources from several sources from a single point [29]. The idea is to use the Spider to search for Widgets and learning content that can be included in a mashup learning environment and then use IMS CC to package them into a "package" that, when unwrapped, constitutes a mashup learning environment. In conjunction to this it seems reasonable to start discussing digital learning resources from a broader perspective – not just being about learning content, but also functional components, such as widgets.

In parallel with the developments described in the previous section, another project will start where some pedagogical experiments will be carried out using the WiMUPPLE environment, where the idea of mashup learning environments will be tested in real pedagogical situations with students and teachers.

REFERENCES

[1] F. Paulsson. Modularization of the Learning Architecture: Supporting Learning Theories by Learning Technologies. Royal Institute for Technology (KTH): Stockholm, 2008. 121.

[2] F. Paulsson and A. Naeve. "Virtual Workspace Environment (VWE): A Taxonomy and Service Oriented Architecture Framework for Modularized Virtual Learning Environments - Applying the Learning Object Concept to the VLE". International Journal on E-Learning, 2006, vol. 5, no. 1, pp. 45-57.

[3] G. Atwell. Personal Learning Environments - the future of eLearning? [online]. 2007, [Retrieved: April, 2012]. Available from World Wide Web: www.elearningeuropa.info/files/media/media11561.pdf

[4] T. Erl. SOA Principles of Service Design. 1 ed. Prentice Hall: Boston, MA, 2007.

[5] F. Paulsson. "A Service Oriented Architecture-framework for modularized Virtual Learning Environments," In A. Mendes-Vilas, A. Solano Martin, J. Mesa Gonzáles, and J.A. Mesa Gonzáles, Current Developments in Technology-Assisted Education. FORMATEX, 2006, pp. 21-62.

[6] S. Wilson, O. Liber, M. Johnson, P. Beauvoir, P. Sharples, and C. Milligan. "Personal Learning Environments: Challenging the dominant design of educational systems," First European Conference on Technology Enhanced Learning (ECTEL), 2006,

[7] D. Jones. "PLES: FRAMING ONE FUTURE FOR LIFELONG LEARNING, E-LEARNING AND UNIVERSITIES". Lifelong Learning Conference, 2008,

[8] D.A. Wiley. "Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy". In D.A. Wiley, The Instructional Use of Learning Objects. Bloominton: Agency for Instructional Technology and Association for Educational Communications & Technology, 2002, pp. 3-23.

[9] F. Paulsson and M. Berglund. "Suggesting a SOA-framework for modular virtual learning environments: comparing two implementation approaches," International Journal of Web-Based Learning and Teaching Technologies, 2008, vol. 3, no. 1, pp. 43-57.

[10] P. Brereton and D. Budgen. "Component-based systems: a classification of issues. Computer", 2000, vol. 33, no. 11, pp. 54-62.

[11] O. Nierstrasz and L. Dami. "Component-Oriented Software Technology," In O. Nierstrasz and D. Tsichritzis, Object-Oriented Software Composition. Hertfordshire: Prentice Hall International (UK) Ltd, 1995, pp. 3-28.

[12] C. Szyperski. Component Software - Beyond Object-Oriented Programming. Translated by C. Szyperski; 2 ed. ACM Press: New York, 2002. 229.

[13] J. Preciado, S. Comai, and C. Sánchez-Figueroa. "Necessity of methodologies to model Rich Internet Applications,". In the proceedings of the Seventh IEEE International Symposium on Web Site Evolution (WSE'05), 2005.

[14] M. Ogrinz. Mashup Patterns Designs and Examples for the Modern Enterprise. Addison Wesley: 2009.

[15] S. Sire and A. Vagner. "Increasing Widgets Interoperability at the Portal Level," The First International Workshop on Mashup Personal Learning Environments (MUPPLE-2008), 2008.

[16] J. Yu, B. Benatallah, F. Casati, and F. Daniel. "Understanding Mashup Development," IEEE Internet Computing, 2008, vol. 12, no. 5, pp. 44-52.

[17] J. Wong and J. Hong. "What do we "mashup" when we make mashups?" In the proceedings of the 4th international workshop on End-user software engineering, 2008.

[18] D.R. Herrick. "Google this!: using Google apps for collaboration and productivity," SIGUCCS '09 Proceedings of the 37th annual ACM SIGUCCS fall conference, 2009, pp. 55-64.

[19] N. Sultan. "Cloud computing for education: A new dawn?" International Journal of Information Management, 2010, vol. 30, no. 2, pp. 109-116.

[20] V. Hoyer and M. Fischer. "Market Overview of Enterprise Mashup Tools," Service-Oriented Computing, ICSOC 2008. In Lecture Notes in Computer Science, 2008, vol. 5364, pp. 708-721.

[21] Fielding, R. T. 2000. Architectural Styles and the Design of Network-based Software Architectures. Information and Computer Science. Doctor of philosophy, 76-106.

[22] F. Paulsson and A. Naeve. "Establishing technical quality criteria for Learning Objects,". in the proceedings of eChallenges 2006, 2006, vol. 2, pp. 451-462.

[23] D. Crockford. The application/json Media Type for JavaScript Object Notation (JSON) [online]. 2006, [Retrieved: April, 2012] Available from World Wide Web: https://tools.ietf.org/html/rfc4627

[24] Yahoo! Query Language [online]. 2011, [Retrieved: March, 2012] Available from World Wide Web: http://developer.yahoo.com/yql/

[25] S. Sire, M. Paquier, A. Vagner, and J. Bogaerts. "A messaging API for inter-widgets communication,". In the proceedings of the 18th international conference on World Wide Web, 2009,

[26] IMS Common Cartridge Specification [online]. 2008, [Retrieved: April, 2012]. Available from World Wide Web: http://www.imsglobal.org/cc/index.html

[27] IMS GLC Learning Tools Interoperability Basic LTI Implementation Guide. 2010, vol. Version 1.0 Final Specification.

[28] C. Severance, T. Hanss, and J. Hardin. "IMS Learning Tools Interoperability: Enabling a Mash-up Approach to Teaching and Learning Tools," Techn, Inst, Cognition and Learning, 2010, vol. 7, pp. 245-262.

[29] F. Paulsson. "Connecting learning object repositories - strategies, technologies and issues". In the proceedings of the Fourth International Conference on Internet and Web Applications and Services (ICIW09), 2009 pp. 583-589.

[30] Weber, N., Nelkner, T., Schoefegger, K., and Lindstaedt, S. N. 2010. SIMPLE - a social interactive mashup PLE. the Third International Workshop on Mashup Personal Learning Environments (MUPPLE09), in conjunction with the 5th European Conference on Technology Enhanced Learning (EC-TEL2010).

[31] Wheeler, S. 2009. Learning Space Mashups: Combining Web 2.0 Tools to Create Collaborative and Reflective Learning Spaces. Future Internet. 1, 1, 3–13. DOI=10.3390/fi1010003.