Client-sided vs. server-sided MUPPLEs: Can add-ons beat platform-based solutions?

Felix Mödritscher[1],

[1] Institute for Information Systems and New Media,
Vienna University of Economics and Business,
Augasse 2-6, 1090 Vienna, Austria
felix.moedritscher@wu.ac.at

Abstract. Typically Mash-Up Personal Learning Environments (MUPPLEs) are based on different theoretical foundations, which lead to different development strategies and architectural styles. In this paper we examine the influence of the two most opposite PLE architectures – server-sided and client-sided solutions – towards relevant social and technical aspects. All in all, we come to the conclusion that the decision for a specific architecture mainly depends on very few factors – precisely collaboration, privacy and trust, computational efforts and scalability, as well as software development and maintenance – and that some shortcomings can be addressed by additional technical concepts, like a practice sharing infrastructure in the back-end or the utilization of existing web-based tools to achieve specific functionality.

Keywords: Personal Learning Environments, Architectural Styles, Social Aspects, Technical Factors, Technology Review.

1 Introduction

About two years ago, Wild et al. (2008) introduced the concept of Mash-Up Personal Learning Environments (MUPPLEs), extending the character of Personal Learning Environments (PLEs) by aspects of End-User Development (Lieberman et al., 2006) and Opportunistic Design (Hartmann et al., 2008). Precisely, a MUPPLE comprises a PLE solution which empowers learners to design and use their environments by utilizing existing tools so that they can connect to learner networks and collaborate on shared artifacts. In this sense, a MUPPLE infrastructure enables users to mash up existing functionality, i.e. in the form of web applications and widgets, to an environment which is useful for them to achieve personal goals and learning needs.

Although being a young concept, various software systems and tools can be classified as MUPPLE solutions, or include at least few aspects of a Mash-Up Personal Learning Environment. Amongst others, Palmer et al. (2009) give an overview of different dimensions for building web-based PLEs, reaching from UI-related issues over data interoperability and social dimensions to runtime behavior of PLEs. Furthermore Palmer and his colleagues evaluate various platforms according to these characteristics. However, each of the PLE software reviewed is realized in the

form of a server-sided platform (e.g. iGoogle, Netvibes, Moodle/Wookie, Google Wave, etc.), and also the dimensions for building web PLEs do not consider other architectural styles than the (web) server-client architecture.

From a more technical perspective, MacNeill and Kraan (2010) elaborate different architectural styles of learning environments and summarize five particular degrees of distribution: (1) one system in the cloud, many outlets; (2) plug-ins to existing virtual learning environments (VLEs); (3) many widgets from the web into one widget container; (4) many providers and many clients; (5) both a provider and a client at the same time. From our point of view, a MUPPLE solution is closely related to the last type of a distributed learning environment, as it contains all tools and devices a learner requires for her everyday activities. In any way, this architectural style does not specify if the learning environment should be realized in the form of a server-sided system or a software tool being executed on the computer of a learner.

In this paper, we compare server-sided MUPPLE platforms to client-sided solutions which have been emerged in the last year – for instance we consider our own prototype, the so-called 'Personal Activity Manager' (PAcMan, see https://addons.mozilla.org/en-US/firefox/addon/176479/) as a client-sided PLE. The comparison of these two architectural MUPPLE styles will be conducted according to different social and technical aspects which are relevant for personal learning environments. Therefore, the following two sections elaborate advantages and disadvantages of server and client-sided MUPPLE technology along social and technical issues. In section 4 we summarize the findings of this comparison and determine which architectural style is better for a MUPPLE solution.

2 Social aspects of MUPPLE architectures

Personal learning environments aim at supporting learners in utilizing different tools in order to participate in collaborative activities of learning communities. Consequently the architecture of a MUPPLE does have an influence on various social aspects, as elaborated in the following.

2.1 Collaborative activities

Van Harmelen (2006) states that a PLE (and therefore also a MUPPLE) aims at empowering learners to design their learning environments in order to connect to learner networks and collaborate on shared artifacts to achieve common goals. Therefore facilities for PLE-based collaboration are considered to be important. As evidenced by many social networking platforms (like Facebook, MySpace, XING, etc.) and Web 2.0 tools (e.g. Wikipedia, Delicious, Flickr, etc.), server-sided web solutions perfectly enable collaboration between users. However, if understanding a PLE as an infrastructure to find and manage these (web-based) applications also client-sided MUPPLEs are fit enough to foster collaboration amongst learners. The main advantage of a server-sided to a client-sided architecture deals with core functionalities of a PLE, as will be explained in the upcoming subsection.

2.2 Sharing and reusing PLE experiences

Mödritscher and Wild (2009) report on practice sharing in PLE settings, outlining the importance of facilities to capture, share, retrieve, and re-use learners' experiences on utilizing (web-based) tools for their different (learning) activities. Such facilities can be built upon data mining mechanisms based on log files, learner interaction recordings, contextual attention metadata, and so forth. In the sense of educational mining (Romero & Ventura, 2006), the semantics extracted from this data can be exploited e.g. for recommending artifacts or tools for a specific situation, finding appropriate peers, visualizing knowledge and important relations, providing navigational aids, etc. In this context, Mödritscher et al. (2010) introduces the concept of a pattern repository for practice sharing which can be plugged into PLE solutions. By enabling learners to share patterns of their activities, learning experiences can be made public and re-used by others. The usage of such an activity pattern can trigger competence development if it fits a learner's situation and is new to her.

Similar to Web 2.0 platforms and approaches (like Facebook, Google, etc.), server-sided MUPPLEs have clear advantages for practice sharing to client-sided solutions. On the other hand, the pattern repository described by Mödritscher et al. (2010) is independent of the architecture, as it is integrated into the PLE via a web-based API. Such a repository could be hosted by different parties, e.g. a university offering alumni learning services, a company trying to foster internal practice sharing, a private organization supporting their members, etc. Using such a back-end infrastructure for sharing good PLE practices would equalize the benefits of server and client-sided MUPPLEs.

2.3 Trust

With respect to social networking sites (cf. Dwyer et al., 2007) a first issue to mention for PLEs is trust. Trust can be understood as "*the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party*" (Mayer et al., 1995). In connection with software, trust is always closely related to the trustworthiness of the software vendor or provider. Thus aspect is relevant for the PLE solution as well as for each tool – web or desktop application! – being part of the learning environment.

However, as PLEs capture critical user data in the form of recordings of learner interactions, trust also depends on the control of a user over this data. Here, users might feel more comfortable if the interaction recordings remain on their computers and can be shared at their request. Independently of the architectural style of a PLE, Dwyer et al. (2007) state that only a trust in the solution provider facilitates information sharing. This issues effects both client and server-sided PLEs. Concerning trust in general, we see slight advantages for client-sided PLE solutions.

2.4 Privacy

Similar to trust, we consider privacy a critical aspect in the scope of Mash-Up Personal Learning Environments. Privacy is defined as "*the interest that individuals have in sustaining a 'personal space', free from interferences by other people and organizations*" (Clarke, 2006). The tools as well as digital recordings of learner interactions are part of the personal space and should be secured to preserve the learner's privacy, which is particularly necessary for open(-content) systems (García-Barrios, 2009). In the scope of PLEs it is important that shared information does not expose sensitive user data and that learners can determine what they want to expose to others, e.g. through adaptable privacy statements. Furthermore García-Barrios (2009) proposes an approach to create privacy awareness by visualizing the effects of privacy statements on user data as well as their benefits within the tools available.

Again, the privacy of software in general depends on the reliability of the vendor or provider, or the community behind it. Addressing the risk of data leakage, server-sided platforms are more vulnerable to unauthorized access on critical data, as they are public targets. Client-sided PLEs, on the other hand, are not that exposed and allow users to secure their data through applying security software (see subsection 3.1). User-adaptable privacy statements are considered to be a feature of the PLE solution thus not dependent on the architectural style.

Privacy awareness concepts are realizable in the form of warnings and automated anonymization techniques on sharing PLE-based interaction recordings. Platform solutions have the advantage that they can mine all user data and include a visualization approach as mentioned above. Such a crowd-sourcing approach can be realized through a back-end repository for publishing and retrieving PLE practices (cf. Mödritscher et al., 2010) but requires more efforts and might not take all interactions into consideration. Therefore we see server and client-sided PLEs in balance here.

3 Technical aspects of MUPPLE architectures

Beside these social aspects, the architectural style of a Mash-Up Personal Learning Environment also has an impact on various technical factors. In the following subsections we briefly sketch these aspects and try to order them according to their importance for realizing MUPPLE solutions.

3.1 Computational efforts and scalability

A first and highly relevant factor to mention is the processing effort for providing all functionality of the personal learning environment. Following the hard facts of large-scale Web 2.0 platforms, a wide-spread PLE solution would require a lot of computational power, as evidenced with the distributed calculation of the Google index (Barroso et al., 2003) or with the many specialized, high-scalable technologies and servers behind Facebook (Royal Pingdom, 2010). Due to the fact that Mash-Up

PLEs aim at integrating existing tools into one's learning environment, a lot of computational effort is outsourced to the servers providing these web applications.

For the remaining PLE infrastructure, a server-sided architecture has clear disadvantages to client-sided MUPPLEs if a Facebook like platform is being targeted. Although computational resources could be used on the client computers (e.g. through JavaScript and Ajax) the server would have to deal with a high computational load and a lot of communication traffic. Parts of these efforts could be shifted to other servers, as we have shown for instance with the practice sharing infrastructure (cf. Mödritscher et al., 2010) but a distributed server architecture is required in any way. On the other hand, a clients-sided MUPPLE would have to serve only one user who can directly work with web applications or make use of back-end services in order to access a sharing infrastructure. Collaboration with peer actors can be fully achieved with the different tools available to and used by the learners. All kinds of computation are performed on the client machine, having a client-sided PLE approach the more scalable solution.

3.2 Software development and maintenance

A second critical issue of PLEs is the one on rolling out and maintaining such solutions. From the perspective of a developer and provider of a PLE product, a server-sided architecture has clear benefits in providing access to a web-based platform and maintaining this centralized system. As mentioned in the last subsection, scalability might require deploying a server cluster in the back-end, which would increase the efforts in maintenance. For accessibility reasons a MUPPLE platform should be also compliant with the mostly used browsers, which however can be achieved by using mature, wide-spread Ajax libraries. Generally server-sided PLE solutions can be easily rolled out within a community or organization, while new features and patches can be installed quickly and at a central point.

On the other hand, client-sided PLEs are restricted to a user's computer, e.g. to an operating system, a browser or even a certain version of a browser. Thus, the complexity of the software developed tends to be low while large-scaled roll-outs can be problematic if users underlie technical restrictions on their computers e.g. due to an organizational ICT policy. For this situation it would be possible to provide pre-configured installation packages or runnable versions e.g. on a memory stick. Maintenance of client-sided PLEs is easily achievable through automated update mechanisms, whereby the updates normally have to be triggered by the users. Consequently installing time-critical patches is problematic for client-sided approaches. Concerning roll-outs and maintenance we see clear disadvantages of client-sided PLE solutions, which we have experienced with the PAcMan prototype realized as a Firefox add-on.

3.3 Security

Thirdly, security plays a role for personal learning environments in general. In the scope of MUPPLEs and this paper, we mainly refer to the low-level concepts of

security, namely to computer security and vulnerability to hackers (cf. Landwehr, 2001). Computer security aims at mechanisms to protect the PLE-related data of users, e.g. through authentication, data encryption, etc. Vulnerability describes the weaknesses of a computer system or software which allow attackers accessing or duplicating data of PLE users.

Computer and particularly data security is necessary for both kinds of PLE architectures, i.e. a user's PLE-related data should be protected from unauthorized access over a web-based interface or to the physical device (a client computer or a server machine). Client computers can be stolen more easily and might not be protected as good as server systems. In case of data abuse however a vulnerability of a server-sided MUPPLE would concern sensitive data of many PLE users. Moreover servers often provide specific services which increase the vulnerability to hackers. Finally server-sided PLEs are also more vulnerable to attacks on the communications between the server and the client computers, i.e. hackers could also exploit security leaks of a browser to break into the server machine. Overall, we consider client-sided MUPPLEs as more secure than server-based platforms.

3.4 Availability

Availability is defined as the degree to which a MUPPLE is operable and accessible by the end users. Hereby it has to be distinguished between the availability of the web-based tools and the PLE infrastructure integrating the tools and providing users a unified access to their learning environments. The availability of the tools has to be guaranteed by the providers responsible for them, thus it is not relevant for a PLE in general. The availability of a PLE infrastructure is highly related to its architecture. While client-sided MUPPLEs are available even if the computer is not connected to the Internet server-based solutions always require a connection to the server hosting the PLE platform. Consequently, client-sided MUPPLEs can be used for offline activities (e.g. designing one's activities and the corresponding environments), as long as no access to one of the web-based tools or background services is required.

3.5 Responsiveness and usability

The responsiveness of a MUPPLE describes how quickly the environment responds to user input. Similar to the availability of a PLE we have to differentiate between the tools being part of the learning environment and the infrastructure. The responsiveness of tools is fully carried by the tool providers and relies on issues like the connection to the hosting server or the usability of the applications itself. The responsiveness of the PLE infrastructure however is influenced by the architectural design. As client-sided solutions do not require continuous communication with a server, they are a clear favorite regarding this factor. Particularly they do not depend on the quality of the Internet connection like MUPPLE platforms.

Usability of a PLE solution highly depends on the responsiveness but can also consider aspects like single-sign on. It is obvious that a PLE solution – independently of the architectural style – will be used less if users have to authenticate for every tool

being part of an environment separately. Here, platform-based MUPPLE require some server-sided mechanism, like OpenID or Shibboleth, to overcome this problem while clients, i.e. browsers, support users by storing authentication data locally. Consequently, working with a client-sided PLE is more comfortable if users are (semi-)automatically logged into the tools they need within their activities.

3.6 Analyzability

According to Mödritscher (2009), an important aspect of PLEs – and semantic technologies in general – deals with analyzing user data and extracting semantics to be utilized for supporting end-users such as lifelong learners within their communities. Therefore it is necessary to capture and bring together the user data which can either be entered explicitly or recorded while interacting with the environment. The tools integrated into one's PLE are normally recording their own log files. At best, one can analyze them through a cross-platform approach, as seen with the PALADIN approach (Klamma et al., 2008).

Server-sided MUPPLEs have the advantage that they capture and store the interaction recordings at a central point, so this data can be analyzed immediately. Client-sided PLE solutions lead to recordings distributed over many client computers. Therefore they always require a back-end service (like the pattern repository mentioned in section 2) for collecting the data from different users, e.g. all participants in one activity. As a consequence, server-sided PLE platforms fulfill the requirements on analyzability of user data, while client-sided MUPPLEs lack facilities to bring together the interaction recordings.

3.7 Visualization and other user interface issues

Concerning usability and visual feedback mechanisms, a PLE solution should enable the application of visualization techniques and comfortable user interface features, like drag&drop functionality, interactive UI elements, etc. Here, client-sided MUPPLEs have certain advantages. For instance browsers provide possibilities to integrate libraries and executables, e.g. to realize fancy visualization and navigation utilities like the Cooliris add-on (cf. http://www.cooliris.com). Moreover Mozilla's XML User Interface Language (XUL, see https://developer.mozilla.org/en/xul) specification allows developers to modify and adapt all existing UI elements of the browser and add state-of-the-art facilities, which we have done within our PAcMan add-on (available at https://addons.mozilla.org/en-US/firefox/addon/176479/).

Server-sided PLE platforms rely on technologies for creating dynamic web interfaces (HTML and JavaScript, Java Applets, Flash, Silverlight, etc.) or, better, on respective libraries, e.g. Ajax frameworks, Flash libraries, and so forth. The consideration of local programs (libraries or executables) is not possible at all. Overall both architectural styles support the provision of usable UI elements and graphical information. The realization is expensive if not utilizing good libraries, however client-sided have slight advantages to MUPPLE platforms concerning this factor.

3.8 Scrutability

A key characteristic in connection with personalization systems is scrutability (Holden & Kay, 1999). As system developers determine "*what can be modeled, how it is represented, how users can contribute to it and how it can evolve*" (Kay, 2000), it is important that users can understand how an adaptive system works and how the user can influence the adaptation process. Thus, user models have to be scrutable in order to guarantee controllability of a personalization system. In MUPPLEs user profiles – referring to García-Barrios (2008) the simplest form of a user model – can be explicitly found within the tools used as a part of the environment.

Certain PLE solutions – such as the MUPPLE.org prototype, a widget-enabled LMS like Moodle, pure widget engines like Wookie or iGoogle, etc. – provide user profile features as part of their core functionality. The higher-level aspects of user models however are considered as a hidden component implicitly contained in the user interactions with the environment (Mödritscher et al., 2010) and externalized in the form of interaction recordings for one of the learner's activities. Addressing the scrutability of a learning environment, the architectural style seems to have no impact on this aspect at first sight. Having a closer look a valuable feedback on the behavior of one's personal learning environment depends on the factors examined in the last two subsections, analyzability and visualization.

4 Overview, findings and concluding remarks

In Table 1 we have ranked the issues examined in the former sections according to their advantages and disadvantages for each of the two architectural styles. Therefore, we have assigned the most critical aspects to the two PLE architectures in a first step. Then, we have ordered the other issues according to their dependency on the critical ones, which is described in the following.

Table 1. Overview of PLE-related aspects, ranked by their relevance for an architectural style.

	Social perspective	***Technical perspective***
Server-sided MUPPLEs	Collaboration (2.1)	Analyzability (3.6)
		Software development/maintenance (3.2)
↑	Sharing/reusing PLE	Scrutability (3.8)
|	experiences (2.2)	Visualization/user interface (3.7)
|		Security (3.3)
↓	Trust (2.3)	Availability (3.6)
Client-sided MUPPLEs		Responsiveness/usability (3.5)
	Privacy (2.4)	Computation/scalability (3.1)

From the perspective of social factors, server-sided MUPPLE architectures are better in terms of collaboration and practice sharing while client-sided solutions have benefits concerning privacy and trust. As collaboration is an important driver of learning and privacy is regulated by law, we weighted these two issues stronger. On a technical level, platforms are ideal for analyzing user data, for rolling out and

maintaining the software, and for explaining adaptation effects to end-users (scrutable system behavior). As argued in the former sections, client-sided MUPPLEs have advantages concerning the distribution of computational efforts, their availability, responsiveness and usability, as well as security and user interface issues.

To refine the ranking we use the similarity and dependency relations between some of these PLE-related issues. For instance, scrutability and providing visual feedback rely on the possibility to analyze user data and visualize valuable semantics extracted from the interaction recordings, e.g. a collaboration graph of an activity or one's immediate learner network. Furthermore availability and responsiveness are very similar, both assumed to be good if no remote connection is required. Additionally, privacy and trust are related to security, as they focus on user data. On the other hand, these three factors are in opposition to analyzability and data mining.

All in all, we think that the decision for a certain architectural style mainly depends on the primary criteria(s) of a MUPPLE solution. Concluding this paper we have identified two important, opposing combinations of (social and technical) aspects: (1) collaboration and analyzability vs. privacy, trust and security, (2) computational efforts and scalability vs. efforts in software development, roll-outs and maintenance. These main challenges might also explain why community-enabling software primarily follows one of the two well-known streams: (a) large-scale platforms like Facebook, MySpace, XING, Google, Twitter, etc. or (b) highly specialized, useful tools realized in the form of add-ons and client-sided programs (e.g. Skype, ICQ, Cooliris, Shareaholic, PAcMan, or any of the 125 million add-ons available at http://addons.mozilla.org/de/firefox/ on August 16, 2010).

Considering the success of platform-based community approaches (like Facebook) it seems that MUPPLE platforms will be the future – however this assumption might fail for PLEs, as privacy and trust are important factors if dealing with sensitive data such as learner interaction recordings or the artifacts produced in (lifelong) learning activities. So far we are more convinced of client-sided PLE solutions which can overcome various shortcomings (analyzability, practice sharing or collaboration) by the application of adequate tools in one's environment and by specialized back-end services like the pattern repository.

Acknowledgement

The research leading to these results has partially received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 231396 (ROLE project).

References

Barroso, L.A.; Dean, J.; Hölzle, U.: Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro*, 23(2), pp. 22-28 (2003)

Clarke, R.: What's Privacy? *Proc. of the Workshop at the Australian Law Reform Commission* (2006), retrieved from http://www.rogerclarke.com/DV/Privacy.html (2010-08-13).

Dwyer, C.; Hiltz, S.R.; Passerini, K.: Trust and privacy concern within social networking sites: A comparison of Facebook and MySpace. *Proc. of Americas Conference on Information Systems (AMCIS)*, August 10-12, Keystone, Colorado (2007)

García-Barrios, V.M.: Personalisation Systems: Multi-purpose User Modelling. Saarbrücken: VDM Verlag (2008)

García-Barrios, V.M.: User-centric Privacy Framework: Integrating Legal, Technological and Human Aspects into User-Adapting Systems. *Proc. of the International Conference on Computational Science and Engineering (CSE)*, August 29-31, Vancouver, Canada, pp. 176-181 (2009)

Hartmann, B.; Doorley, S.; Klemmer, S.R.: Hacking, Mashing, Gluing: A Study of Opportunistic Design and Development. *IEEE Pervasive Computing*, 7(3), pp. 46-54 (2006)

Holden, S.; Kay, J.: The scrutable user model and beyond. *Proc. of the AIED Workshop on open, interactive, and other overt approaches to learner modelling*, July, Le Mans, France, pp. 51-62 (1999)

Kay, J.: User modeling for adaptation. In C. Stephanidis and G. Salvendy (Eds.): *User Interfaces for All*. Human Factors Series, Lawrence Erlbaum Associates, pp. 271-294 (2000)

Klamma, R.; Spaniol, M.; Denev, D.: PALADIN: A Pattern Based Approach to Knowledge Discovery in Digital Social Networks. *Proc. of the International Conference on Knowledge Management (I-Know)*, September 6-8, Graz, Austra, pp. 457-464 (2006)

Landwehr, C.E.: Computer security. *International Journal of Information Security*, 1(1), pp. 3-13 (2001)

Lieberman, H.; Paterno, F.; Klann, M.; Wulf, V.: End-User Development: An Emerging Paradigm. In H. Lieberman, F. Paterno, and V. Wulf (Eds.): *End-User Development*. LNCS, vol. 4321, Dordrecht: Springer, pp. 1-8 (2006)

MacNeill, S.; Kraan, W.: Distributed Learning Environments: A brief paper. *JISC Centre For Educational Technology and Interoperability Standards (CETIS)* (2010), retrieved from http://wiki.cetis.ac.uk/images/6/6c/Distributed_Learning.pdf (2010-08-13)

Mayer, R.C.; Davis, J.H.; Schoorman, F.D.: An Integrative Model of Organizational Trust. *The Academy of Management Review*, 20(3), pp. 709-734 (1995)

Mödritscher, F.: Semantic Lifecycles: Modelling, Application, Authoring, Mining, and Evaluation of Meaningful Data. *International Journal of Knowledge and Web Intelligence (IJKWI)*, 1(1/2), pp. 110-124 (2009)

Mödritscher, F.; Petrushyna, Z.; Law, E.L.-C.: Utilising Pattern Repositories for Capturing and Sharing PLE Practices in Networked Communities. *Proc. of the International Conference on Knowledge Management (I-Know)*, September 1-3, Graz, Austria, pp. 150-161 (2010)

Mödritscher, F.; Wild, F.: Sharing Good Practice through Mash-Up Personal Learning Environments. *Proc. of the International Conference on Web-based Learning (ICWL)*, August 19-21, Aachen, Germany, pp. 245-254 (2009)

Palmér, M.; Sire, S.; Bogdanov, E.; Gillet, D.; Wild, F.: Mapping Web Personal Learning Environments. *Proc. of the MUPPLE Workshop at the European Conference on Technology Enhanced Learning (EC-TEL)*, September 29, Nice, France, pp. 31-46 (2009)

Romero, C.; Ventura, S.: Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, 33(1), pp. 135-146 (2007)

Royal Pingdom: Exploring the software behind Facebook, the world's largest site. *Pingdom AB* (2010), retrieved from http://royal.pingdom.com/2010/06/18/the-software-behind-facebook/ (2010-08-13)

Van Harmelen, M.: Personal Learning Environments. *Proc. of the IEEE International Conference on Advanced Learning Technologies (ICALT)*, July 5-7, Kerkrade, The Netherlands, pp. 815-816 (2006)

Wild, F.; Mödritscher, F.; Sigurdarson, S.: Designing for Change: Mash-Up Personal Learning Environments. *eLearning Papers*, Vol. 9 (2008)