elearningpapers

Designing for Change: Mash-Up Personal Learning Environments

Fridolin Wild, Felix Mödritscher and Steinn Sigurdarson
Institute for Information Systems and New Media,
Vienna University of Economics and Business Administration

Summary
Institutions for formal education and most work places are equipped today with at least some kind of *tools* that bring together *people* and content *artefacts* in learning *activities* to support them in constructing and processing information and knowledge. For almost half a century, science and practice have been discussing models on how to bring personalisation through digital means to these environments. Learning environments and their construction as well as maintenance makes up the most crucial part of the learning process and the desired learning outcomes and theories should take this into account. Instruction itself as the predominant paradigm has to step down.

The learning environment is an (if not 'the') important outcome of a learning process, not just a stage to perform a 'learning play'. For these good reasons, we therefore consider instructional design theories to be flawed.

In this article we first clarify key concepts and assumptions for personalised learning environments. Afterwards, we summarise our critique on the contemporary models for personalised adaptive learning. Subsequently, we propose our alternative, i.e. the concept of a mash-up personal learning environment that provides adaptation mechanisms for learning environment construction and maintenance. The web application mash-up solution allows learners to reuse existing (web-based) tools plus services.

Our alternative, LISL is a design language model for creating, managing, maintaining, and learning about learning environment design; it is complemented by a proof of concept, the MUPPLE platform. We demonstrate this approach with a prototypical implementation and a – we think – comprehensible example. Finally, we round up the article with a discussion on possible extensions of this new model and open problems.

Keywords: Personalised Learning; Environments; Design, LISL, MUPPLE approach

1 Introduction: Most Important Concepts, Key Arguments, Structure of the Article

Within this article, we are looking back at this history of personalised, adaptive learning to formulate a critique on the contemporary models and theories, while at the same time proposing a new approach which puts learners centre stage again. We will argue that this approach is more apt to explain adaptive personalisation in technology-enhanced learning and is more helpful in guiding (even end-user driven) engineering and maintenance of personalised learning environments. The approach we propose has been developed within the scope of the European IST project 'iCamp' (Kieslinger et al., 2006).

In particular, we are going to show that *learning environment design* is the missing link, able to avoid the flaws of prior adaptation theories in technology-enhanced learning. This is strongly based on three assumptions.

First, we assume that *learning to learn* while at the same time learning content is a better approach than just (re-)constructing domain-specific knowledge. In other words, we believe that the acquisition of social, self, and methodological competence (i.e. transcompetences) prior to or in addition to professional competence is superior to only acquiring professional competence (i.e. domain-specific skills, facts, rules, and the like). This is not only justified through the added value, but additionally by the decreasing half-life of professional knowledge. We deliberately say 'constructing' as in constructivist theory a 'transfer' of knowledge does not exist: knowledge can only be created from within the minds of the learners, though of course influenced on sensory experiences provided by their environment.

Second and consequently, we presuppose that establishing a *learning environment*, i.e. a network of people, artefacts, and tools (consciously or unconsciously) involved in learning activities, is part of the *learning outcomes*, not an instructional condition. Adaptation strategies go beyond navigational adaptation through content artefacts along a predefined path: for example, some learners may prefer to email an expert instead of reading an online paper. Adaptation has to take place along individualised activities performed in these environments. Inherently, these learning environments are always networks: they encompass actors, artefacts, and tools in various locations with heterogeneous affiliations, purposes, styles, objectives, and the like. Network effects make the network exponentially more valuable with its growing size.

Third and finally, we consider *emergence* of behaviour as an unavoidable and natural phenomenon of complex networks. By emergent behaviour we mean that the observable dynamics show surprising activity, surprising in so far as the participating systems have not been instructed to do so (they may even not have intended it). Designing for emergence is in our view more powerful than 'programming' by rules as the models involved are simpler while achieving the same effect.

Based on these assumptions, we are going to sketch a new model for personalised adaptive learning which strongly focuses on learning environment design. We discuss representational aspects of this model by proposing a learning interaction scripting language with which environment design can be formalised. Furthermore, we demonstrate the feasibility and illustrate our approach with two examples performed with our research prototype 'mupple'.

The rest of this paper is organised as follows. First, we summarise our critique on the contemporary models for personalised adaptive learning. Then, we propose our alternative, i.e. the concept of a mash-up personal learning environment that provides adaptation mechanisms for learning environment construction and maintenance. Third, we demonstrate our approach with a prototypical implementation and a – we think – comprehensible example. Finally, we round up this article with a discussion on possible extensions of this new model and (still) unresolved problems.

2 Personalisation: Why Instructional Design Theories Fail and Why Current Adaptation Technologies are Defective by Design

Classically, the field of personalised adaptive learning bases on **instructional design theories** and utilises **adaptive and intelligent technologies** for personalisation.

Instructional design theories aim at offering explicit guidance to help people learn better and, consequently, they treat learning environments (the tools!) as an instructional condition and separate from the desired learning outcomes (cf. Reigeluth, 1999). Even in more constructivist instructional theories, the learning environment is assumed to be created by an instructional designer (Mayer, 1999; Jonassen, 1999). In applied research, these design theories appear in two different flavours: with and with-out a strong artificial intelligence component.

Theories incorporating a strong AI viewpoint are inherently ill-defined as they need to take into account all context variables that may influence the learning process of the learners. Invested in this approach, however, is a naive objectivist assumption that it is possible to create an omniscient artificial system which knows everything (or in a weaker form 'everything important') about the current context variables influencing a learner in his information processing and learning work. This is not

possible. Learners are not sitting in a glass-box where a teacher can monitor which Wikipedia pages they are reading, to whom they are talking in the hallways, and whether their childhood experiences influence them to prefer reading to watching television. Even *iff* a learner would have lived his whole life in a glass-box, still it would not be possible to distinguish the relevant from the irrelevant environmental influences and the resulting representational model(s) would have the same complexity as the original learner. By itself it already would require an infinite amount of adaptation work, even growing exponentially with the number of people participating in a learning network. And still – like in Searle's famous Chinese room –, even if a system could number-crunch a problem of this complexity, it would never truly understand what the learner is thinking.

Contemporary instructional design theories, however, have abandoned this goal of a strong artificial intelligence monitoring and guiding automatically a long time ago. Usually, they foresee a mixture of minor automatic system adaptations along a coarse-grain instructional design master plan engineered by a teacher or instructional designer. So-called 'learning paths' are fine-tuned along learner characteristics and user profiles to conform to trails envisioned, not necessarily proven by teachers.

There are two good reasons, why these weak AI theories have to be rejected, when applied for personalisation. First (and less important), there is no 'perfect' instructional designer: an environment can only be planned for the average learner, not the individual. Even good instructional designers had to gain their experience, had to make errors in the past to built up effective and efficient strategies. Moreover, in practice instructional designers are most often 'only' domain experts for a particular field of knowledge, no didactical experts. Second (and more important), planned adaptation takes experiences away from the learners: external planning keeps them from becoming competent, as it takes chances to self-organise away and personal discovery is prevented. Learners, however, are not only sense-makers instructed by teachers along a predefined path. Learners need to actively adapt their learning environment to their needs so that they can construct the competences necessary for successful learning. And facilitators can coach them on this way.

The learning environment is an (if not 'the') important outcome of a learning process, not just a stage to perform a 'learning play'. For these good reasons, we therefore consider instructional design theories to be flawed. Learners are not patients that need an aptitude treatment. They proactively have to (and of course already do) take account of their learning environment.

Adaptation technologies can vary in their degree of control: how much are end-users (learners!) involved in adaptation. Oppermann, Rashev, and Kinshuk (1997) therefore differentiate between adaptive and adaptable systems with a fluent transgression from the one to the other. Systems are considered to be adaptable if the users initiate the adaptation (and vice versa). Similarly, Dolog identifies two perspectives through which adaptation can be seen: adaptations can be performed by humans to cope with changed requirements of the participating stakeholders. Alternatively, adaptation can be a dynamic system adaptation to changed parameters in the environment or context (Dolog, 2008).

Three important streams of research can be identified as relevant for personalised adaptive learning: technologies from *adaptive (educational) hypermedia*, *learning design technologies*, and *adaptive hypermedia generators*.

On a finer level, *adaptive and intelligent technologies* can be distinguished into curriculum sequencing and problem-solving support technologies (Brusilovsky, 1999). Whereas sequencing deals with adapting the navigational path through pre-existing learning material, problem-solving support technologies deal with evaluating the student created content representations either summatively or – in interactive support technologies – formatively even during the learning process itself through the provision of feedback or by presentation of related examples. Furthermore, in the more generic adaptive hypermedia area, adaptive navigation support and adaptive presentation support can be distinguished (Brusilovsky, 1999). Both deal with adapting pre-existing content: navigation support with path and link adaptation (though in a more open setting – the web), the latter with the presentation of a subset in new arrangements of the available content. Additionally, a third

class of approaches is mentioned by Brusilovsky (1999), which deals with student model matching: they try to make use of collaborative filtering aspects (either by identifying matching peers or by identifying differences to avoid problems).

Henze and Brusilovsky (2007) identify the lack of reusability and interoperability as a major problem in personalised adaptive learning. When applying adaptation in the web, this results in the 'open corpus problem' which can (at least partially) be compensated by gaining more interoperability. For adaptation interoperability, however, standards are still missing (Henze and Brusilovsky, 2007; Kravcik, 2008).

Holden and Kay (1999) postulate that scrutability has to become a key characteristic in personalisation strategies: evidence *accreated* (i.e. collected) from various sources is *resolved* (i.e. assessed) at request time while providing control over the input as well as output streams and providing inspection capabilities for the processing mechanisms. Though this offers triggers for reflective activities, these activities are not part of the modelled user activities. They merely are performed outside the system, thereby neither supported nor hindered by the system.

Although these adaptive (educational) hypermedia technologies all differ, they share one characteristic: they deal primarily with the navigation through content, i.e. the represented domain specific knowledge. Information processing and knowledge construction activities are not in the focus of these approaches. Consequently, they do not treat environments as learning outcomes and they cannot support learning environment design.

Koper and Tattersell (2005) state that in their *learning design* introduction they will be using 'learning design' synonymously with 'instructional design', though there may be a slightly different accent in the meaning of both. Specht and Burgos (2007) elaborate on the adaptation possibilities in general and particularly within IMS-LD. However, among the generic components of educational systems that can be adapted, they list only pacing, content, sequencing, and navigational aspects. Neither does the environment (not even tools) appear in this list, nor is it a driving factor for information gathering, nor method of adaptation (Specht and Burgos, 2007). Towle and Halm (2005) discuss how adaptive strategies can be embedded with units of learning by filtering or reordering resources, changing methods, slotting learners into roles (and scaffolding role transitions), or by changing activities. Van Rosmalen and Boticario (2005) investigate how – besides design time – also run-time adaptation can be realised with LD, thereby interfacing LD with distributed multi-agent systems. They tweak LD to incorporate agents (by adding them as 'staff'). Though, adaptation of the environment only takes place to a very limited degree: The aLFanet project does not foresee to help in managing a complex set of tools and services out of an even more complex, not determined portfolio at run time. Olivier and Tattersall (2005) explain the possibilities of integrating learning services in the environment section of LD. Besides the already mentioned restriction that these components both in practice and in the guidelines are only touched by instructional designers, the services are postulated to be known at design time: they are approved in the specification (LD 1.0 has four services!), and additionally they have to be instantiated through formal automated procedures. From the perspective of standardisation, Olivier and Tattersall predict application profiles that enhance LD with the services provided by particular communities, though interoperability to other LD players then no longer will be given. Within the TENcompetence project, extensions have been proposed that allow for more bottom-up oriented authoring of the units of learning (Vogten et al., 2008): formalisation, reproducibility, and reusability of learning designs can also be catalyzed through the use of a personal competence manager that facilitates the development of learning materials through learners themselves.

Though in principle people, activities, artefacts, and services (not tools!) are constituting components of LD, the standard does not offer support for communication and reflection on technology use on a higher granular level, nor facilitates environment building and maintaining. Moreover, LD is based on the assumption that the services that can be deployed in an environment have to be shared by all executing software players. Hereby, 'services' differ in so far from 'tools', as tools relate additionally to the perceivable surface of a learning network. Both interface human activity with machine communication (i.e. digital thus manipulable information). However, tools also incorporate their user

interfaces and their design influences the processes pursued with them, as has been shown for example by Pituch and Lee (2004). Additionally, agreement on the standardisation of services can always only be a second step after innovating new services. We have to state that also current LD (together with the available authoring tools and players) fails to support competence achievement in learning environment design, also is the environment set-up and maintenance not part of the learning activities.

A third block of research can be identified in the group of *adaptive hypermedia generators* (Ceri et al., 2005). Cristea, Smits, and De Bra (2007) report on LAG, a language used to express information on assembly, adaptation, and strategies plus procedures of intelligent adaptation applications. It was developed specifically for adaptive educational hypermedia. LAG follows the structure of hypertexts and expresses rule-based path adaptation (the 'adaptation dynamics') for automatically adapting course contents. WebML in combination with UML-Guide has been deployed to realise client-sided adaptation of e-learning web-applications (Ceri et al., 2005). WebML follows a generic hypertext model and contains the structural elements of site views, areas, pages, and content units. A site view is a hypertext consisting out of areas which again can integrate sub-areas and pages. Pages are the actual containers for information to be given to the user. They consist out of elementary content units that extract data with queries from data sources. By combining it with slightly extended UML state diagrams (UML-Guide), user navigation through a system can be modelled, and – through both – personalised applications can be generated (Ceri et al., 2005). Though in principle not restricted the environmental aspects of a typical design process are recommended to be executed by a designer rather than a learner. The environment design itself, however and again, is restricted to content and path design.

Though especially the generator technologies could take account of more recent developments on end-user development (the long tail of software development, cf. Lieberman et al., 2006), all of the approaches are focused on the classical instructional design paradigm: learners are executing along minor adaptations what instructional designers (mostly teachers) have foreseen. Consequently, emergence does not play a significant role in these approaches. It is 'rule', not 'environment'!

3 The Concept of a Mash-Up Personal Learning Environment

Considering the learning environment not only a condition for but also an outcome of learning, moves the learning environment further away from being a monolithic platform which is personalisable or customisable by learners ('easy to use') and heading towards providing an open set of learning tools, an unrestricted number of actors, and an open corpus of artefacts, either pre-existing or created by the learning process – freely combinable and utilisable by learners within their learning activities ('easy to develop'). Often the generic direction of research behind this is called end-user development (Lieberman et al., 2006).

In this section we describe the development of a technological framework enabling learners to build up their own personal learning environments by composing web-based tools into a single user experience, get involved in collaborative activities, share their designs with peers (for 'best practice' or 'best of breed' emergence), and adapt their designs to reflect their experience of the learning process. This framework is meant to be a generic platform for end-user development of personal learning environments taking into account the paradigm shift from expert-driven personalisation of learning to a design for emergence method for building a personal learning environment. In the following we introduce our approach to learning environment design, consisting out of a **learner interaction scripting language (LISL)** and its prototypical implementation called **Mash-UP Personal Learning Environment (MUPPLE)**.

Repenning and Ioannidou (2006) elaborated that a language for end-user development has to consider thirteen design guidelines (DG1 to DG13). Each of these guidelines is referenced in the upcoming paragraphs whenever it is relevant for a design decision. Following DG4, i.e. to "make domain-oriented languages for specific end-user development", we specified the so-called **Learner Interaction Scripting Language (LISL)** to empower learners with capabilities to construct and maintain

their personal learning environment: they can specify actions and interactions with others, with artefacts, and with tools.

The development steps and important design principles of this personal learning environment design language are highlighted in the following.

First of all, LISL has to reflect the semantics of its underlying, generic pedagogical model, thereby following DG5 to "introduce meta-domain orientation to deal with general end-user development". To be independent of a subject domain, we applied activity theory in a similar way as manifested for the INCENSE system (Akhras and Self, 2000). Thus, we derived a simplified learner interaction model shown in Figure 1. Basically, we break down the learning context into situations which describe the physical and social environment of learners. In such a situation, a learner is engaged in a so-called activity which consists of actions and includes tools, artefacts, and other actors (facilitators or peers). In contrast to instructional design, these actions represent more prominently commands for self-organising the learning process.

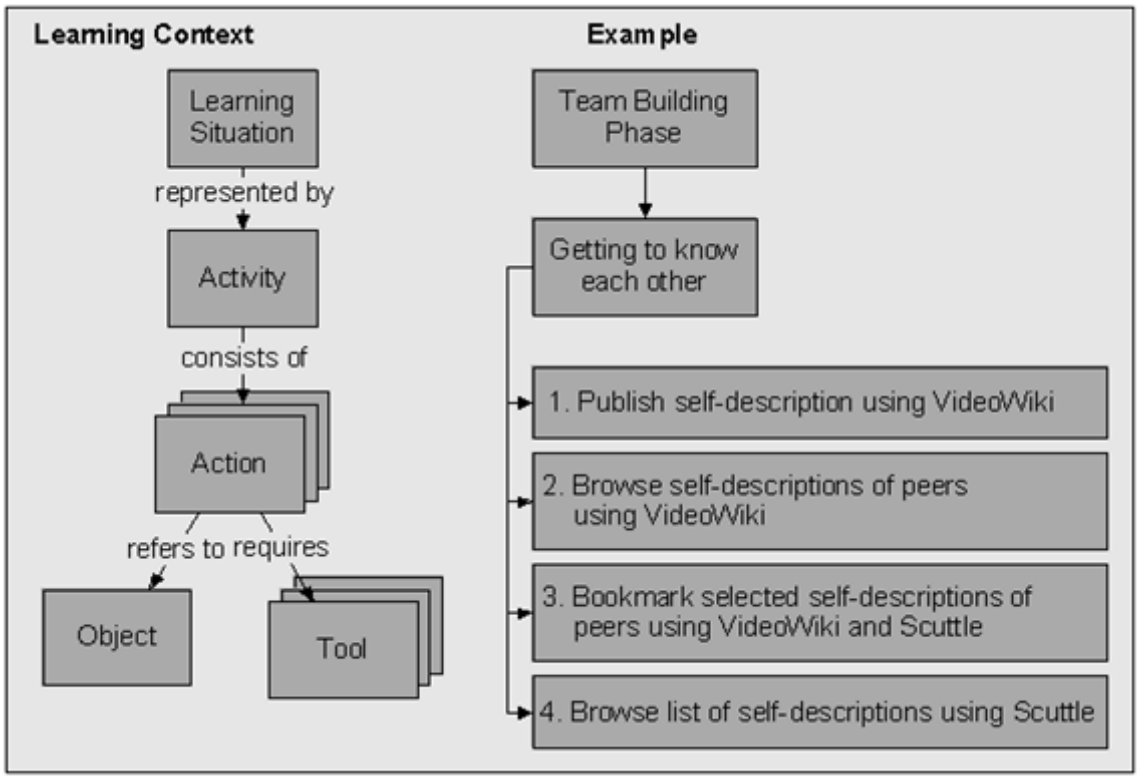

Figure 1. Semantic model behind MUPPLE, including the exemplary activity 'Getting to Know Each Other'.

An activity consists of list of actions which the learner can perform sequentially, utilising tools for an action. Such a learning activity is meant to be our basic entity in which learners make their experiences and develop competence actively. Moreover, this notion of a learning activity is simple and understandable for learners, thereby helping to direct and scrutinise systemic behaviour (Kay, 2000). Learners use the language constructs to define actions and artefacts, integrate new tools, and perform what they planned by following the self-instruction coded in action statements, expected outcomes, and objectives.

To enable reflective learning (Boud, Keogh, and Walker, 1985), we decided to bind each action to one specific object (artefact) and at least one tool in order to produce one outcome. Although different actors can work on the same action and also produce outcomes, each learner primarily sees his own behaviour and results. Moreover, these action-object-tool triples are recommended to peers when defining and executing new actions. Collaborative as well as coexisting LISL scripts form a learning network of actors, artefacts, and activities (Koper, Rusman, and Sloep, 2005): the MUPPLE platform thereby serves as a meeting point (DG13: "build community tools").

In the field of usability engineering (cf. Nielson, 1993), learnability deals with the easiness for novice users to accomplish basic tasks on encountering a design for the first time while efficiency refers to how quick expert users can perform tasks. LISL allows experienced users to script their learning activities efficiently, while novices use the web-based widgets and dialogues of MUPPLE for creating their learning environment, which is materialised into LISL statements by the platform. LISL has a natural language like syntax, so progressing to the level of an expert should be relatively quick for novices.

The 'easy to learn' objective of LISL empowers learners not only to write their own scripts, but additionally to modify existing ones shared by others: LISL is considered extensible, it satisfies the postulate of DG6 to "support incremental development". Furthermore, LISL is designed for actions, objects, and tools to be personal, as requested by DG11: "allow immersion".

This support LISL has for exchangeability of learning scripts, means it facilitates the emergence of best-of-breed solutions, i.e. by learners creating activity patterns from their own learning activities, sharing them with others who in turn may modify and re-share to "scaffold typical designs" (DG12).
The example learning activity 'Getting to know each other' mentioned in the last figure is used to explain the most important concepts of LISL. In the first three lines of Figure 2 three actions are defined, whereby 'publish' and 'bookmark' have fixed URLs (a REST call for creating a page within the Wiki tool and the call for bookmarking a Wiki page). On the other hand, 'browse' comprises an action which can be used for different tools, so a specific URL is not needed here.

```
[1] define action "publish" with url http://videowiki.icamp.eu/addvideo?namespace=gtkea
[2] define action "browse"
[3] define action "bookmark" with url http://scuttle.icamp.eu/bookmarks.php?action=add[...]
[4] define action "self-description"
[5] define action "selected self-description"
[6] define action "all self-descriptions" with url http://videowiki.icamp.eu/[...]/namespace/gtkea
[7] define action "my list of self-descriptions" with url http://scuttle.icamp.eu/mylist
[8] define tool "videowiki" with url http://videowiki.icamp.eu
[9] define tool "scuttle" with url http://scuttle.icamp.eu
[10] publish "self-description" using videowiki
[11] browse "all self-descriptions" using videowiki
[12] bookmark "selected self-description" using videowiki
[13] browse "my list of self-descriptions" using scuttle
```

Figure 2. *LISL code for the exemplary learning activity 'Getting to know each other'.*

Concerning the four objects used in this scripting example (lines 4 to 7), two of them ('all self-descriptions', 'my list of self-descriptions') are defined as static objects with a fixed URL. The other two ('self-description', 'selected self-description') are not specified at all, which means they represent a reference (placeholder) for self-descriptions created or selected by a user. Thus, patterns using this LISL scripts have to be additionally personalised by learners in the way that these unspecified objects reflect their individual values. The lines 8 and 9 define the two web-based tools used in this learning activity, namely the 'VideoWiki' tool and the social bookmarking tool 'Scuttle'. Both are expected to have their own URLs that are used in MUPPLE if no action or object URL is given.

Finally, the last four lines of this LISL code example describe the actions of this activity. Thereby, 'publish' is a typical action being associated with one object ('self-description') and one tool ('videowiki'), prescribing that the learner should create a self-description with that tool. Furthermore, 'browse' is used twice, once for going through the self-descriptions of the peers and again for browsing the list of collected self-descriptions. For both actions the VideoWiki tool is recommended. The 'bookmark' action, however, is more complicated, as it requires VideoWiki to work together with Scuttle in the way that a user can send links of VideoWiki recordings (selected self-descriptions) to the bookmarking tool. This kind of 'channel' between these two tools has to be realised through an API in the backend, e.g. by implementing a REST-based API or using a feed management API such as FeedBack (Wild and Sigurdarson, 2008) to base the channel on a feed.

Analogous to the field of compiler construction, we consider our web-based prototype **MUPPLE** to be a runtime environment for LISL scripts. Thus, the execution of LISL code 'runs' the personalised web application thereby realising the semantics of a learning activity expressed in the scripts. This runtime environment consists of three parts: (1) the activity model which already was addressed in the last subsection, (2) an integrative infrastructure for web-based tools, and (3) the LISL interpreter.

In the technological infrastructure for the presentation layer of MUPPLE, we build upon the web 2.0 paradigm of mash-ups, more specifically in our case a web-application mash-up. While a mash-up in the traditional sense combines data streams from multiple sources into an integrated user experience, a web-application mash-up is more focused on aggregating user interfaces together with data. A solution approach like our XoMashup component (cf. Mödritscher et al., 2008) has to consider certain issues, as explained in the following.

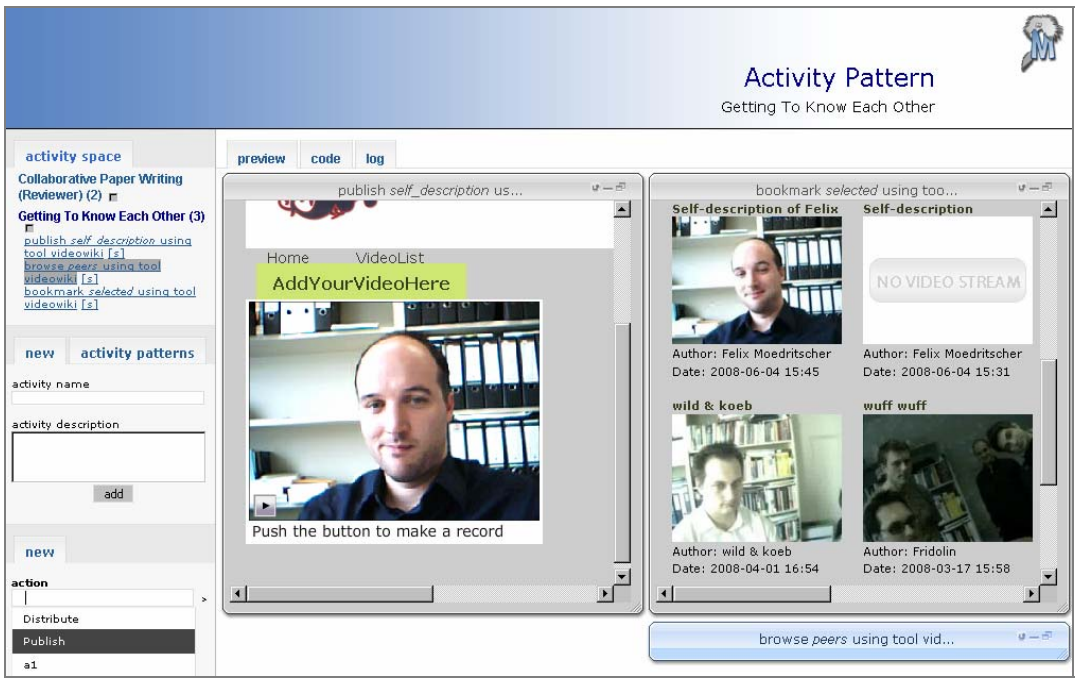

Figure 3. *Personal learning environment generated by MUPPLE for the learning activity 'Getting to know each other': header (top), control elements (left), and content area (middle).*

Concluding from mash-up visualisation techniques (Spoerri, 2007), displaying different applications next to each other requires some scaffolds for users to reduce the cognitive load while working with the system. Similar to iGoogle, MyYahoo, Netvibes, and other providers of personalised websites, we realised a portal-like OpenACS component which allows users to arrange tools along a layout grid (see also content area in Figure 3). The tools and objects presented in the windows are again part of the language elements (cf. DG3, "use objects as language elements"). Every action-object-tool triple in fact a quadruple, as it (theoretically) also includes the subject of this instruction: the person who builds this personal learning environment. Action statements can always be seen as from a ego perspective: 'I (should) publish this self-description'. This "encourage[s] syntonicity" (DG10).

In sum, the web application mash-up solution allows learners to reuse existing (web-based) tools plus services. It forms the technological infrastructure for our MUPPLE approach. Although we still face technological restrictions like a lack of system interoperability, web application mash-ups are very flexible and, therefore, useful for other application areas.

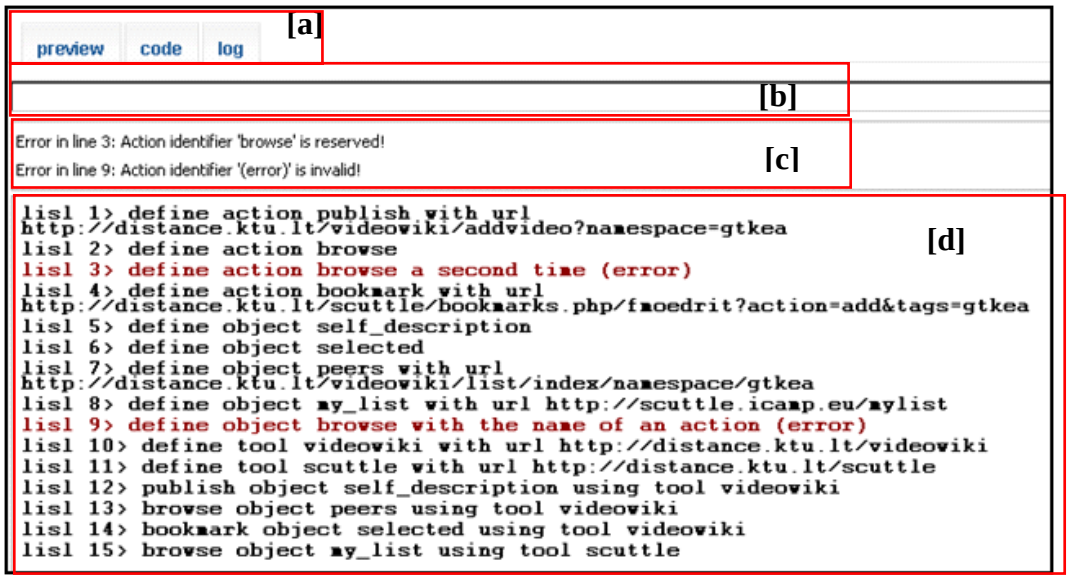

Figure 4. *Output of the LISL interpreter for the exemplary activity 'Getting to know each other', including errors (line 3 and 11).*

Next to the mash-up preview, our first prototype also includes an interpreter shell for the LISL scripts and guiding click-through dialogues to produce extensions, as requested by DG9, "integrate development tool with web service". As shown in Figure 4, section [a], tabs are used to switch between the mash-up space (preview), the interpreted LISL code of the current MUPPLE page (log), and a code authoring mode (code) to "Provide multiple views with incremental disclosure" (DG8). In the interpreter shell (log) users may enter new LISL statements, and test their way along which "facilitate[s] decomposable test units" (DG7). If the entered LISL statements do not throw errors, they are appended to the LISL code of the page, which helps a bit to "make syntactic errors impossible" (DG2). Figure 4, section [c] also demonstrates how invalid statements within the persisted script are visualised, as is indicated with the errors in line 3 and 9, which helps at least a bit to "make syntactic errors hard" (DG1).

Finally, LISL code is also added to the page content if the user is working with the control widgets in the preview-mode. Here, changes of the semantic model, e.g. creating a new action or moving on to another one, and user changes of the visual appearance, like dragging portlets to another position in the lay-out grid, are simply appended to the end of the script.

As a personal learning environment design language, LISL supports the full life-cycle of users' interactions in their learning environments, reaching from defining activity models over specifying action steps, involved or produced artefacts, collaboration with others, and the deployment of tools.

Additionally, best-of-breed sharing is supported. Particularly for attracting new users, the success of MUPPLE also depends on recognisable benefits for learners. Therefore we encourage sharing of activity scripts, to facilitate for the emergence of best-of-breed patterns. Similar to the idea of scripting collaborative activities (Dillenbourg and Jermann, 2007), the LISL scripting language is able to describe these activity patterns. Thus, learners can export parts of their learning scripts in order to publish activity patterns and consume learning experiences of others by creating learning activities from available patterns.

The network effect dictates that the value of being in the network increases exponentially with the number of participants, or connected nodes to the network. MUPPLE leverages the network effect first and foremost by encouraging the sharing and modifications of existing learning patterns, additionally the system aims to provide learners with increased benefits from the network by recommending widely used patterns, tools and actions. These recommendations are not only interesting from an adaptive perspective, but also from a social network perspective, as they provide a surface to the network which may trigger change in learner behaviour. In this respect, the

recommendations themselves are not important, rather the way they may produce a network effect on learner behaviour.

To support learners in defining and executing own actions, the bottom-up approach of MUPPLE also allows automatic analysis of former learning scripts in order to personalise different aspects of learning. So far, we implemented a service for recommending action-object-tool triples to inexperienced learners. However, the LISL scripts can be understood as user profiles distributed over the activities a learner is involved in. Therefore, it is also thinkable that these profiles are analysed automatically and reasoning on learning behaviour is conducted, so that additional support can be provided to learners.

4 Example: Collaborative Paper Writing

To illustrate the utilisation of the design language we introduced in the previous section, the following part describes an in-depth scenario and an outline of how the personal learning environment constructed in this scenario can be reflected with a LISL script. For this example we will assume that some of our authors are using MUPPLE as their LISL platform. Reaching beyond the before-mentioned activity 'Getting to Know Each Other', we focus on a typical activity in the field of higher education: collaborative paper writing.

In the following scenario, a learner wants to collaborate with experts from other organisations in writing a paper. With a few clicks, she creates a personal learning environment for this activity that consists out of six steps encompassing actions on identifying and sharing literature, subsequently summarizing the state of the art with the help of this literature, distributing chapter assignments to the collaborators, and finally elaborating the text of the assigned parts. She benefits from earlier users of the system who already configured several of the tools she is going to use, most notably Scuttle, a social bookmarking tool: MUPPLE already knows how particular actions can be executed in specific tools and which specific url's address these actions. Without really noticing, as she is using the guiding menus of the graphical user interface, MUPPLE added a couple of additional lines to her new activity script (see Figure 5). Most of the lines 1-5 and 6-11 have been added by the system automatically according to her intended use of the actions that are reflected in lines 12-18 of the depicted script.

As the famous peer-to-peer paper search engine ObjectSpot was not known to the system, she added the tool conveniently while specifying the new action 'find' with the object 'literature' in the dialogues.

```
 [1] # tools
 [2] define tool ObjectSpot with url http://www.objectspot.org/
 [3] define tool Scuttle with url http://distance.ktu.lt/scuttle/
 [4] define tool XoWiki with url http://xowiki.icamp.eu/
 [5] define tool WebMail with url http://webmail.wu-wien.ac.at/

 [6] # configure tools for certain actions
 [7] define action bookmark for Scuttle with url
       http://distance.ktu.lt/scuttle/bookmarks.php/?action=add&tags=literature
 [8] define action browse for Scuttle with url http://distance.ktu.lt/scuttle/tags.php/literature
 [9] define action summarize for XoWiki with url http://xowiki.icamp.eu/state-of-the-art
[10] define action elaborate for XoWiki with url http://xowiki.icamp.eu/paper
[11] define action review for XoWiki with url http://xowiki.icamp.eu/review

[12] # actions
[13] find literature using ObjectSpot
[14] bookmark literature using Scuttle
[15] browse bookmarks using Scuttle
[16] assign chapters using WebMail
[17] summarize "state of the art" using XoWiki
[18] elaborate paper using XoWiki
[19] review paper using XoWiki
```

Figure 5. LISL script for the activity 'collaborative paper writing'.

She decides to share this activity as a pattern with her collaborators, so that they can instantiate it and customise it to their needs. Subsequently, each participant of the planned paper writing activity

instantiates his activity from the pattern and personalises it by adding, removing, or reconfiguring actions contained in the pattern script depending on their own relevance judgements for this task. The first author probably uses all actions, particularly the action on 'assigning chapters' as her role is to manage the activity. Co-authors might focus more on identifying and sharing literature and elaborating the paper parts.

One of them later-on discovers that a review could benefit the writing project, so he adds an additional action to 'review' the 'paper' (see line 19 of Figure 5).

Now most of the co-authors start simultaneously collecting bookmarks on the papers they found in ObjectSpot. As they add these bookmarks to Scuttle one by one and as the script foresees that they all use the same tag 'literature' (see line 7), the portlet displaying the bookmark lists (initiated by line 15) fills slowly with data aggregated from all participants. They start summarizing the state of the art from these bookmarks using the XoWiki page started by line 17. At a point in time, the manager has the impression that she can distribute the writing work of the core piece and informs her collaborators about their assignments (see line 16) using a webmail client. Later-on all elaborate the core piece using the tool deployed by line 18. As it is a wiki, versioning is guaranteed and collaboration facilitated. A final review by one of the collaborators initiates another revision cycle till the paper is finished.

Figure 6 displays the mash-up of web applications MUPPLE creates when executing the LISL script of Figure 5.

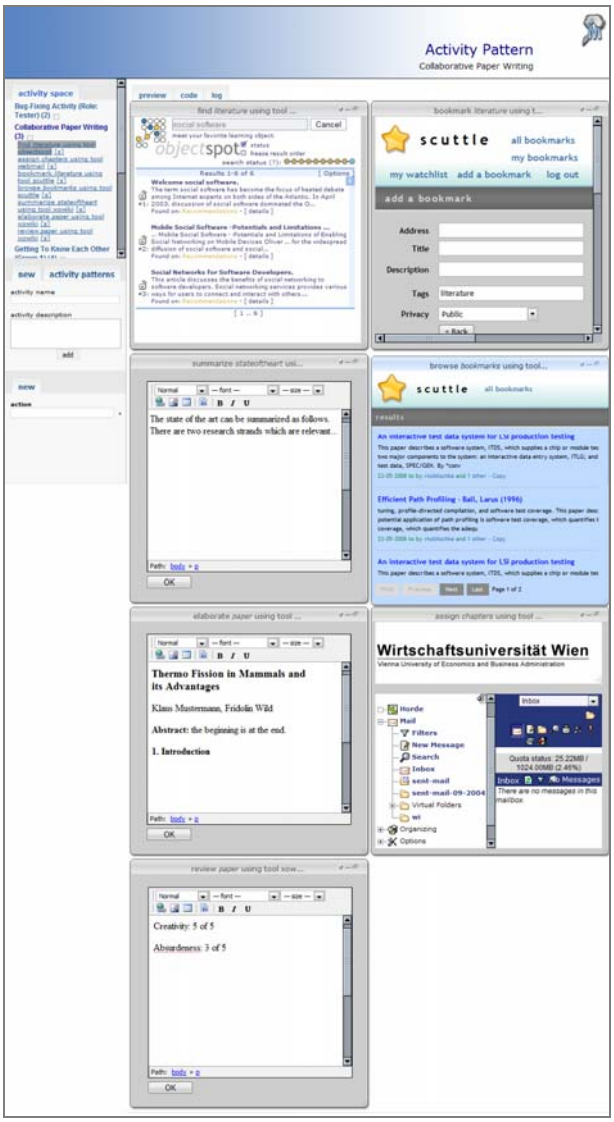

Figure 6. The rendered script in MUPPLE.

5 Conclusions and Outlook

Hannafin, Land, and Oliver (1999) have stressed the fact that with the expected growth of both information and technology, new models for instructional design have to be sought: an information-age paradigm of instruction. Given our analysis of today's instructional design theories and adaptation technologies, we go even further: learning environments and their construction as well as maintenance makes up the most crucial part of the learning process and the desired learning outcomes and theories should take this into account; instruction itself as the predominant paradigm has to step down.

Classical instructional design theories assume that the environment should more or less automatically adapt to the user. But it should be the other way around: the user should easily adapt the environment to her needs. It is not about learning design it is all about learning environment design. Managing distributed cognition (Poirier and Chicoisne, 2006) through manipulating the learning environment is a key competence for successful learning in the web 2.0.

Within this contribution we have proposed our alternative: LISL is a design language model for creating, managing, maintaining, and learning about learning environment design. It is complemented by a proof of concept, the MUPPLE platform. In our new approach, we particularly tried to avoid the problematic aspects of expert-driven, content-model based, instructional adaptation strategies. Personalisation of the learning process takes place through customisation of the learning environment, network effects on collaborating with peers, and recommendations and support given by MUPPLE. It may comprise a new generation of personalised learning environments, the future will show.

Despite of the possibilities and strengths of our MUPPLE approach, a few disadvantages have to be outlined here. Primarily, these problems concern technological issues. First of all, it could be even nicer to have a high degree of interoperability between web applications, which now is not always the case. This specifically relates to single-sign-on procedures and communication channels to transfer both data and events from one application to the other. For example, XoWiki as well as our webmail client require authentication, so learners right now have to log-in separately in each application (not that they were not used to it). Regarding communication channels, we have proposed (Wild and Sigurdarson, 2008) a specification how to realise distributed feed networks with buffered-push capabilities. We intend to further investigate these means and will see how they can be incorporated into LISL (maybe with additional 'connect' statements between tools). We can think of other approaches, though, and we do not have a solution for the efficient communication of events.

Secondly, the utilisation of iframes causes problems in cross-domain scripting (cf. Jackson and Wang, 2007). Finally, we are also aware that LISL and MUPPLE still lack important functions, especially in the area of regulating collaboration and privacy.

References

Akhras, F.N., and Self, J.A. (2000). System Intelligence in Constructivist Learning. *International Journal of Artificial Intelligence in Education*, 11(4), pp. 344-376.

Aroyo, L., Dolog, P., Houben, G-J., Kravcik, M., Naeve, A., Nilsson, M., and Wild, F. (2006). Interoperability in Personalized Adaptive Learning. *Educational Technology & Society*, 9(2), pp. 4-18.

Boud, D., Keogh, R., and Walker, D. (Eds.) (1985). *Reflection: Turning Experience Into Learning*. Abingdon, OX: Routledge.

Brusilovsky, P. (1999). Adaptive and intelligent technologies for web-based education. *Künstliche Intelligenz*, 4/99, pp. 19-25.

Ceri, S., Dolog, P., Matera, M., and Nejdl, W. (2005). Adding Client-Side Adaptation to the Conceptual Design of e-LearningWeb Applications. *Journal of Web Engineering*, 4(1), pp. 21-37.

Cristea, A., Smits, D., and De Bra, P. (2007). Towards a generic adaptive hypermedia platform: a conversion case study. *Journal of Digital Information (JoDI)*, 8(3).

Dillenbourg, P., and Jermann, P. (2007). Designing integrative scripts. In Fischer, F., Mandl, H., Haake, J., and Kollar, I. (Eds.): *Scripting computer-supported collaborative learning: Cognitive, computational and educational perspectives*. New York: Springer, pp. 277-302.

Dolog, P. (2008). Designing Adaptive Web Applications. *SOFSEM 2008: Theory and Practice of Computer Science*, Berlin: Springer, pp. 23-33.

Hannafin, M., Land, S., and Oliver, K. (1999). Open Learning Environments: Foundations, Methods, and Models. In C.M. Reigeluth (Ed.): *Instructional Design Theories and Models*. Mahwah, NJ: Lawrence Erlbaum Associates, pp. 115-140.

Holden, S., and Kay, J. (1999). The scrutable user model and beyond. *AIED Workshop W7: Open, interactive, and other overt approaches to learner modelling*, pp. 51-62.

Jackson, C., and Wang, H.J. (2007). Subspace: Secure Cross-Domain Communication for Web Mashups. *Proceedings of the International Conference on World Wide Web (WWW)*, pp. 611-620.

Jonnassen, D. (1999). Designing Constructivist Learning Environments. In C.M. Reigeluth (Ed.): *Instructional Design Theories and Models*, Mahwah, NJ: Lawrence Erlbaum Associates.

Kay, J. (2000). Stereotypes, Student Models and Scrutability. *Proceedings of the International Conference on Intelligent Tutoring Systems (ITS)*, pp. 19-30.

Kieslinger, B., Fiedler, S., Wild, F., and Sobernig, S. (2006). iCamp: The Educational Web for Higher Education in an Enlarged Europe. In P. Cunningham, and M. Cunningham (Eds.): *Exploiting the Knowledge Economy: Issues, Applications, Case Studies*. Amsterdam: IOS Press, pp. 1440-1448.

Koper, R., Rusman, E., and Sloep, P. (2005). Effective Learning Networks. In: *Lifelong Learning in Europe*, 1, pp. 18-27.

Koper, R., and Tattersall, C. (2005). *Learning Design*. Berlin: Springer.
Kravcik, M. (Ed.) (2008). Current and Future Perspectives for Personalized Adaptive Learning. *Deliverable D1.13 of the Prolearn Network of Excellence*.

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

Mayer, R. (1999). Designing Instruction for Constructivist Learning, In C.M. Reigeluth (Ed.): *Instructional Design Theories and Models*, Mahwah, NJ: Lawrence Erlbaum Associates.

Mödritscher, F., Neumann, G., García-Barrios, V.M., and Wild, F. (2008). A Web Application Mashup Approach for eLearning. *Proceedings of the OpenACS and .LRN Conference*, pp. 105-110.

Nielson, J. (1993). *Usability Engineering*. San Diego, CA: Academic Press.

Olivier, B., and Tattersall, C. (2005). The Learning Design Specification. In: R. Koper, and C. Tattersall (Eds.): *Learning Design*, Berlin: Springer.

Opperman, R., Rashev, R., and Kinshuk (1997). Adaptability and Adaptivity in Learning Systems. In Behrooz (Ed.): *Knowledge Transfer*, II, London: Pace, pp. 173-179.

Pituch, K., and Lee, Y. (2006). The influence of system characteristics on e-learning use. *Computers & Education*, 47(2006), pp. 222-244.

Poirier, P., and Chicoisne, G. (2006). A Framework for Thinking about Distributed Cognition, In Harnad & Dror (Eds.): *Special Issue on Distributed Cognition, Pragmatics & Cognition*, 14(2), Amsterdam/Philadelphia: John Benjamins Publishing.

Reigeluth, C. (1999). *Instructional Design Theories and Models.* Mahwah, NJ: Lawrence Erlbaum Associates.

Repenning, A., and Ioannidou, A. (2006). What Makes End-User Development Tick? 13 Design Guidelines. In H. Lieberman, F. Paterno, and V. Wulf (Eds.): *End-User Development*, Dordrecht: Springer, pp. 51-86.

Specht, M., and Burgos, D. (2007). Modeling Adaptive Educational Methods with IMS Learning Design. *Journal of Interactive Media in Education*, 2007/08.

Spoerri, A. (2007). Visual Mashup of Text and Media Search Results. *Proceedings of the International Conference Information Visualization*, pp. 216-222.

Towle, B., and Halm, M. (2005). Designing Adaptive Learning Environments with Learning Design. In R. Koper, and C. Tattersall (Eds.): *Learning Design*, Berlin: Springer.

Van Rosmalen, P., and Boticario, J. (2005). Using Learning Design to Support Design and Runtime Adaptation, In R. Koper, and C. Tattersall (Eds.): *Learning Design*, Berlin: Springer.

Vogten, H., Koper, R., Martens, H., and Van Bruggen, J. (2008). Using the Personal Competence Manager as a complementary approach to IMS Learning Design authoring. *Interactive Learning Environments*, 16(1), pp.83-100.

Wild, F., and Sigurdarson, S.E. (2008). Distributed Feed Networks for Learning. In: *The European Journal for the Informatics Professional (UPGRADE)*, 9(3).

Authors

Fridolin Wild
Scientist
Institute for Information Systems and New Media
Vienna University of Economics and Business Administration
fridolin.wild@wu-wien.ac.at

Felix Mödritscher
Scientist
Institute for Information Systems and New Media
Vienna University of Economics and Business Administration
felix.modritscher@wu-wien.ac.at

Steinn Sigurdarson
Researcher
Institute for Information Systems and New Media
Vienna University of Economics and Business Administration
steinn.sigurdarson@wu-wien.ac.at

Copyrights

Edition and production

Name of the publication: eLearning Papers
ISSN: 1887-1542
Publisher: elearningeuropa.info
Edited by: P.A.U. Education, S.L.
Postal address: C/ Muntaner 262, 3º, 08021 Barcelona, Spain
Telephone: +34 933 670 400
Email: editorial@elearningeuropa.info
Internet: www.elearningpapers.eu