E-learning frameworks: facilitating the implementation of educational design patterns

Michael Derntl*
Faculty of Computer Science, University of Vienna, Rathausstrasse 19/9, A-1010 Wien, Austria
E-mail: michael.derntl@univie.ac.at
*Corresponding author

Rafael A. Calvo
School of Electrical and Information Engineering, Building J03, University of Sydney, NSW 2006 Australia
E-mail: rafa@ee.usyd.edu.au

Abstract: Educational design patterns hold the promise of facilitating the design of educational activities and environments. We have a number of completed initiatives and projects today, but none of them has had sustained impact on the practitioner community. In this paper, we argue that the use and usability of educational design patterns for technology-enhanced learning can be increased by complementing patterns with ready-to-use tools as part of an e-learning framework approach. We present two different approaches of conceiving such an e-learning framework:

1. an ‘extension’ approach that provides pattern-based, custom LMS components
2. a ‘facade’ approach that enables the pattern-based reuse of existing LMS components.

To demonstrate the feasibility and benefits of e-learning frameworks we present implementations of both approaches.

Keywords: educational design; design patterns; technology-enhanced learning; e-learning frameworks; learning management systems; LMS; beehive; pattern manager; PatMan.


Biographical notes: Michael Derntl obtained his PhD in Computer Science from the University of Vienna. He is a Lecturer and a Researcher at the Faculty of Computer Science, University of Vienna. He has published extensively on topics related to educational technology in international journals, books and conference proceedings. His current research interests include educational design, particularly the role of people, design languages, tools and standards for technology enhanced learning, as well as Web 2.0 and social software for learning and knowledge management.
1 Introduction

Teaching practitioners are facing major challenges when adopting cutting-edge technology in their courses (McNaught et al., 2009). Creative minds keep inventing and developing new tools and applications that are adopted by the current student generation at an incredibly fast pace. Traditional approaches to designing and delivering education are too rigid and inflexible when it comes to adapting to those new trends and adopting new tools and theories. Twenty-first century education calls for more flexible approaches and technologies that succeed in facilitating teachers and teaching practitioners in designing and delivering education on one hand, and in satisfying increased educational quality demands of our students on the other hand.

Considering this background, designing and delivering technology-enhanced education is becoming increasingly complex, particularly for novice designers and teachers. Put very simply, they need to transform educational objectives into executable learning designs. This includes proper selection, definition and delivery of learning tasks and tools to achieve intended learning outcomes (Goodyear, 2005). This process is becoming increasingly difficult to master given the vast array of tools and technologies currently available: technology is leaping forward at an incredible pace, constantly producing new ways of supporting collaboration, interaction, personalisation and delivery (Sigala, 2007) – think of weblogs, wikis, social networking sites, publishing sites, content syndication, to name a few.

Within the scope of this paper, we essentially see the task of the educational designer as one of effectively integrating these technologies into educational processes to enhance student learning experiences.

Other disciplines have a longer tradition or culture of formalising and supporting the design process (Waters and Gibbons, 2004). For instance, software designers have a de-facto lingua franca for describing and formalising their designs – i.e., the Unified Modelling Language (Rumbaugh et al., 1999), architects use agreed-upon symbols and notations to create their blueprints, and so forth. Design languages are one of many tools used by designers in the design process; to support designers in creating high-quality designs, they need expertise and/or advice on how to combine the tools and artefacts to create a work piece (Botturi and Stubbs, 2007). Even though there are several design languages and design methodologies available for technology enhanced learning, few have been adopted by practitioners as part of their daily work and exchange with colleagues (Botturi et al., 2006). A promising methodology to facilitate the dissemination of good design practice for education is design patterns. A pattern captures a recurring
design problem and the essential bits and pieces of the solution in a way that enables reuse by practitioners (Alexander et al., 1977).

Design patterns for education have been around for several years now, especially in the research community, but adoption of patterns by the practitioner community is still very rare. In this paper, we argue that one of the reasons for this lack of adoption may lie in the missing built-in support during instantiation of design patterns. In software engineering, this issue was remedied by providing patterns and tools through application frameworks (Fayad et al., 1999). We have adopted and adapted this approach to the domain of technology-enhanced education and framed a concept we call e-learning frameworks. This paper presents the underlying basics as well as different methods of providing and implementing e-learning frameworks.

The paper is structured as follows. The next section describes the concept of e-learning frameworks, and how they increase reusability following ideas taken from software engineering. Sections 3 and 4 describe the problem of integrating design patterns into LMS and propose two different approaches of approaching this problem. Section 5 presents implementations of the two approaches, and Section 6 concludes the paper with a summary and an outlook on further research directions.

2 Educational design patterns

In the field of architecture, Christopher Alexander was the first to capture, describe and disseminate expert design advice in the form of design patterns. Essentially, a design pattern provides a generic, reusable solution to a recurring design problem or situation. The key is to describe the solution in a way that makes the solution reusable for similar problems (Alexander et al., 1977). Alexander described patterns at varying scope, an example of a small-scope pattern being ‘things from your life’, which deals with what you should place on your cupboards or pin to your walls. For instance, the problem here is an all-white wall; the generic solution to this problem as offered by the pattern is to use things from your life such as pictures of family, souvenirs from trips, etc. The pattern does not prescribe a specific solution (e.g., put a full-size photograph of your dog on one of your living room walls), instead it offers advice on what characteristics a good solution will embody and leaves the implementation details of a concrete solution to be defined by the person applying the pattern. A pattern on a very large scale, for instance, would be ‘agricultural valleys’, which deals with creating a self-sufficient co-existence of cities and towns on hilltops and farmland for crops in the valleys. These examples demonstrate that design patterns can be described and applied on very different levels of granularity and scale.

The design pattern approach was adopted in the 1990s by software engineers to describe elements of reusable object-oriented software design (Gamma et al., 1995; Johnson, 1992) and software architecture (Buschmann et al., 1996). Not much later, computer science educators began to transfer the concept of design patterns to the educational domain (Bergin, 2002; Fincher, 1999; Pedagogical Patterns Project, 2002). There are different flavours of design patterns in education, some more pedagogy focused like patterns for seminars (Fricke and Völter, 2000), other more technology focused like patterns for designing learning management systems (LMS) (Avgeriou et al., 2003). In the recent past, we have witnessed a number of design pattern initiatives and projects dealing with educational design patterns, e.g.:
The Pedagogical Patterns Project (2002) was one of the pioneers; it produced a number of pattern collections related to various aspects of teaching and learning, e.g., patterns for seminars (Fricke and Völter, 2000), active learning, experiential learning, feedback, etc.

E-LEN (E-LEN Project, 2003), a European project providing patterns for learning resources and LMS, life-long learning, collaborative learning, and adaptive learning.

Kaleidoscope (Kaleidoscope Network of Excellence, 2008) is a network of excellence, partly dealing with the provision of design patterns for games-based mathematics education (Winters and Mor, 2008).

TELL (TELL Project, 2005) was an international project that created patterns for networked collaborative learning.

The PCeL pattern repository (Derntl, 2006) provides patterns for implementing technology-enhanced learning based on a person-centred educational philosophy.

One of the main drawbacks of most, if not all, existing design pattern approaches for technology-enhanced learning is the missing support and scaffolding for pattern instantiation – that is, actual pattern usage. Compared to the vast array of patterns available the number of detailed reports, case studies, and evidence of actual pattern use in educational environments is very small. Most projects terminate their efforts right after the production and dissemination of patterns. Also, they focus on reusing design, but not actual tools supporting the implementation of patterns, which is an essential factor in designing for technology-enhanced learning. This is actually similar to what happened in engineering. But in engineering, as the awareness of the need to reuse both design and functionality increased, and after a decade of using design patterns, software engineers started developing application frameworks: tools that combine the two types of reuse (Johnson, 1997). A similar approach has been proposed recently for education (Turani and Calvo, 2006; Turani et al., 2005) by providing design patterns paired with runtime tools that support the patterns. In this paper, we will go one step further and present different methods and implementations of providing application frameworks for e-learning.

There are several considerable differences between applying design patterns for software design and applying design patterns for educational design. For instance, when designing a new piece of software, there is typically a high degree of flexibility in terms of architecture, design and implementation; that is, engineers have many options when choosing, selecting and implementing designs based on patterns. From the software user’s point of view, the design that rests within a piece of software typically is an intangible and invisible artefact. In education, the story is somewhat different; the design of the instruction, the selection and adoption of (existing) tools and systems is highly transparent to the users, i.e., the learners. Particularly in technology-enhanced learning, educators may face a plethora of constraints when designing and implementing their courses, i.e., institutional requirements, the curriculum, or the set of available LMS and tools (McNaught et al., 2009). In technology-enhanced educational design, the selection and configuration of tools and services is a potentially complex task. This is where previous educational design pattern approaches have failed to provide adequate support for their pattern users. We believe that this is an essential aspect for promoting the use of educational design patterns among educators. This paper picks up on this issue by
proposing a framework-based approach to providing and integrating educational design patterns into existing LMS.

3 E-learning frameworks

E-learning frameworks aim to improve the reusability of design and implementation by following the practices originated in software engineering. Within these well established practices, software components are built so they can be reused by a programmer who does not need to know how the component is implemented, and only needs to know its interface, i.e., how the functionalities the component provides can be accessed. This is done by what engineers call ‘abstractions’. Components should be clear-cut and decoupled from each other, yet easy to connect with each other to make new systems that are customisable and effective both functionally and from a design perspective.

In education, practitioners reuse learning materials mostly in an unstructured and manual way, or at best by reusing ‘learning objects’ from institutional or public repositories. LMS also provide tools that allow students to perform common tasks (e.g., participating in a discussion or submitting an assignment), allowing teachers to incorporate these functionalities into their learning designs. In both cases, systems allow teachers to reuse objects or activities, without prescribing a design, or even scaffolding the design process. Teachers must redesign the overall learning activity from scratch for every new course; this adds extra time demands to the already stressed teacher. Theoretically, there are remedies available for these problems, e.g., learning design specifications like IMS LD (IMS Global, 2003), however, practically they are cumbersome to use and barely adopted and supported by popular LMS.

Engineers have brought together both types of reuse (design and implementation) in the form of application frameworks. Although their definition varies, a common definition is that a framework provides the skeleton of an application that can be customised by the application developer. In the case of e-learning frameworks, instead of an application developer we have learning designers or teachers planning the application, i.e., a course or activity.

The pattern integration can be built as an extension to the LMS, where all the components are provided by the LMS and managed by unified pattern-based interface. Another approach is to have a framework built as a separate tool, and then integrated into one or more LMS. The components in this case do not need to be inside the LMS, but can be external. Both approaches are described in detail the next section.

4 Approaches for providing e-learning frameworks

As an example of the ways to integrate an e-learning framework into an LMS, let us consider the case of virtualising ‘project-based learning’: this pattern describes a high-level pedagogy making use of complex, iterative problem solving processes in the context of authentic student projects. The pattern is completed by applying related pedagogical strategies such as teamwork, online learning journals, and other support activities. Now consider we want to apply the ‘luxury’ technology-enhanced version of this pattern using project work supported by virtual team workspaces, asynchronous online communication facilities for teams, online diaries for logging project activities,
and web-based evaluation procedures for self evaluation and peer evaluation of projects. Currently, the problem is that there is little or no support of putting the patterns into practice using an LMS. The pattern user is left alone figuring out how to support the environment suggested by the patterns be configured using the tools available on his/her LMS.

We identify two basic approaches for supporting the building of such an activity on existing LMS using an e-learning framework. The two approaches are described in the following sub-sections.

4.1 The ‘extension’ approach – provision of tailored LMS components

The idea underlying this approach is that e-learning frameworks come with software components that implement the patterns and are optimally integrated or pluggable into an existing LMS. The major advantage of this approach is that it potentially enables optimal support for the patterns, as the components can be tailored to provide a precise implementation of one or more patterns. The disadvantage is that each new pattern potentially requires the implementation of a new LMS component. In addition, for smooth integration, the components need to be plugged into existing LMS, which is only possible under the circumstance that the LMS provides:

a an open application programming interface (API) for integration or

b some form of a plug-in architecture.

Examples of such LMS include Moodle (http://moodle.org), Sakai (http://sakaiproject.org) or dotLRN (http://dotlrn.org). Following this approach, a framework might for instance present a solution to organising virtual discussion groups and include a software component for the LMS that implements the features required by this pattern.

Figure 1 Provision of tailored LMS components as part of the e-learning framework
A general illustration of this method is shown in Figure 1. The LMS offers a set of components or tools \( T = \{t_1, \ldots, t_n\} \), for instance an assignment tool, a chat tool, and so forth. The LMS also offers an extension mechanism (e.g., an open API or plug-in architecture); this is indicated in the figure by the open bridge connecting the LMS with the LMS extensions. In this case, the framework offers a set of patterns \( P = \{p_1, \ldots, p_n\} \) coupled with a set of LMS extensions \( X = \{x_1, \ldots, x_n\} \) enabling the instantiation of the patterns within the target LMS, i.e., framework \( F = P \cup X \). The functionality of the LMS is extended by \( X \), i.e., the set of tools offered to the LMS users becomes \( T^* = T \cup X \). Thereby, each tool may implement several patterns and vice versa (indicated in Figure 1 with dotted connectors).

4.2 The ‘facade’ approach – configuration of existing LMS components

This approach, instead of providing custom LMS components, takes the existing LMS package as given and provides support for configuring available LMS components to implement one or more patterns, optimally in the form of tools (e.g., wizards) that facilitate the instructor during instantiation of a pattern. The major advantage of following this approach is that it does necessarily require the development of additional LMS components as part of the framework. Disadvantages include that patterns/frameworks often require features which are not available on a given LMS. In such cases, the framework would have to use workaround solutions to achieve its goals. As an example, consider a framework suggesting team-based peer evaluation of project documents. Optimally, the LMS would offer links to peer evaluation forms directly from the document folders of the project teams. However, this is usually not offered by current LMS, so that we would often need to implement a workaround, e.g., uploading peer evaluations as attachments onto a discussion board. We called this method ‘facade’ approach since it follows the facade design pattern (Gamma et al., 1995), which solves the problem of using the components of a complex system by introducing a facade component with a unified, easier-to-use interface to a family of client components. In the case of patterns as the ‘client’ components and the set of LMS tools as the target system, the features and functionality required by patterns needs to be mapped to the functionality offered by LMS tools. This is managed by the facade component.

Figure 2 Provision of a facade component to as part of the e-learning framework
An illustration of this approach is given in Figure 2. The framework provides a set of patterns \( P = \{ p_1, \ldots, p_n \} \) and a facade component on top of the existing LMS, i.e., framework \( F = P \cup \{ \text{facade} \} \). Since the interfaces, components or tools provided by the LMS may be incompatible with requirements or features set forth by a specific pattern, the role of the facade component is to help the user instantiate a pattern using the set of existing LMS tools. Hence, the facade defines a left-total relation \( R \subseteq P \times T \), mapping each pattern with at least one tool. For instance, if the user wants to instantiate pattern \( p_2 \), the facade would ‘know’ that the existing tools \( t_1 \) and \( t_3 \) best support this pattern, and it would thus offer a configuration wizard to configure \( t_1 \) and \( t_3 \). The wizard user interface uses terms and concepts used by \( p_1 \) and strives to ‘hide’ the configuration details of \( t_1 \) and \( t_3 \) as much as possible.

5 Integrating educational design patterns into LMS

In the following section, we present a working and publicly available implementation for the ‘extension’ approach in Section 5.1 and a prototypical implementation of the ‘facade’ approach in Section 5.2.

Figure 3  Screenshot of the (partial) tasks page of the group nomination technique
5.1 **Beehive – an example of the ‘extension’ approach**

Beehive (Turani and Calvo, 2006; Turani et al., 2005), is an application framework, combining pre-built software components into patterns, setting default values for those components, and allowing the designer/teacher to modify these values.

Beehive was implemented after modelling research-based pedagogical techniques (brainstorming, debate and jigsaw). These techniques require common tasks to be performed, such as forming groups, provisioning topic information, voting, group discussion, etc. These tasks where implemented as ‘task’ components that can be reused to form many other well-known pedagogical techniques (group nomination, group discussion, round-table discussion, role playing, etc.). These tasks are implemented using commercial off the shelf (COTS) software components that can be automatically mapped to these tasks, hiding the technical difficulty from instructional designers. In our case, all these ‘resource’ components are provided by Flash collaboration server.

Beehive implements 14 design patterns that are customisable sequences of tasks. The tasks repository includes the 29 common tasks organised in five categories. Figure 3 shows some of these patterns in the designer’s interface for configuring the pattern. These tasks can all be implemented with combinations of 12 resource components and access to standard LMS APIs. The resource components include static info viewer, dynamic info viewer, text chat, whiteboard, shared pointer, timer, voting, audio, video, file uploader, and slide presenter.

The tasks sequence has to be described in a language that ‘scripts’ the activity (sequence, timing, etc.) Beehive uses an XML learning design schema based on the IMS learning design specification (IMS Global, 2003).

The advantage of this approach is that it can be more easily integrated to any LMS, as long as they provide APIs for authentication, access control, etc. The disadvantage is that it has to provide a basic set of ‘core’ components, some of which might already be available in the target LMS. Beehive was built using Adobe Flash Communication Server, and integrated to dotLRN and Sakai LMS.

5.2 **PatMan – an example of the ‘facade’ approach**

PatMan (short for ‘pattern manager’) is a tool that was implemented on top of the CEWebS (Mangler and Derntl, 2004) LMS that has been developed at the Faculty of Computer Science, University of Vienna.

PatMan acts as a facade component for the instructor/administrator, helping him/her to set appropriate configuration options for the LMS components that are needed to implement a particular pattern or set of patterns. The underlying idea is to hide the technological details of an instance of a pattern on CEWebS. PatMan has been based on the person-centred e-learning pattern repository (Derntl, 2006) and was implemented as a prototype supporting the application of the following patterns on CEWebS (in alphabetical order): Blended evaluation, collect feedback, course, diary, feedback forum, online discussion, peer evaluation, preliminary phases, project-based learning, project-based learning course, project milestone, questionnaire, reaction sheets, self evaluation, team building, and team workspaces.
As shown in Figure 4, the main page of PatMan for a particular course is divided into four sections:

1. **Currently active patterns**: This section provides an overview of the current course’s structure, containing a list of currently active patterns (i.e., those that have previously been instantiated) in the course. The pattern instances are displayed hierarchically, which means that a pattern that was instantiated as a ‘child’ pattern of some other pattern is listed below its ‘parent’. For instance, ‘peer evaluation’ (child) was instantiated by the ‘blended evaluation’ pattern (parent). The latter supports the instructor during design and implementation of a blend of evaluation strategies during the assessment phases of the course. Currently active patterns can be viewed and edited during design and runtime.

2. **Instantiate a pattern**: This section shows a dropdown box for choosing a pattern to instantiate. The user is then directed to the selected pattern’s instantiation wizard, which guides the user through the necessary configuration steps. Essentially, the pattern instantiation wizards each perform the role of adapter components by offering a user-friendly, stepwise approach of presenting and configuring pattern features, which the PatMan subsequently translates into tool features offered by the LMS. As specified in the presentation of the ‘Facade’ approach in Section 4.2,
PatMan actually ‘masquerades’ and abstracts the configuration options of available LMS tools by providing a meta-configuration layer focusing on actual design practice rather than available LMS component functionality.

3. **Web form manager**: Used to create web forms, which are used for user interaction, e.g., for surveys, peer or self evaluations, and so forth.

4. **Participants data**: This is used to manage LMS specific data and options related to LMS users (students, instructors, tutors, etc.)

Not using PatMan, the instructor would need to activate and configure all necessary LMS components manually based on his/her course design. The screenshot demonstrates that following the ‘facade’ approach adopted by PatMan, the instructor only needs to configure the details of desired activity patterns like student diaries, project milestones, and so forth. All necessary LMS components are then configured automatically by PatMan.

As an example of employing the ‘peer evaluation’ pattern in the final stages of the ‘project-based learning’ pattern, consider Figure 5: the instructor is provided with options for entering his/her preferences of instantiating the pattern. At this stage, the tools that become active on the LMS during the peer review period are ‘concealed’ from the instructor. In fact, after this simple one-step wizard is completed by clicking the ‘Finish’ button, the PatMan actually triggers quite some action within the LMS: it publishes the properly configured evaluation form to the LMS, and creates and configures a link to the evaluation form in each team’s project workspace during the period specified for the peer evaluation activity. So instead of having to manually set these low-level LMS component options, the instructor only needs to take care of peer evaluation options by completing the pattern instantiation wizard.

**Figure 5** Instantiating the ‘peer evaluation’ pattern using PatMan

Following this approach, the PatMan enables a fully pattern-driven instantiation of LMS components, which increases the usability and the intuitiveness of putting educational designs into practice on LMS. A drawback, of course, is that in order to support multiple LMS, the PatMan would have to be implemented separately for each LMS, redeveloping the framework to the target LMS and its components.
6 Conclusions

This paper aimed to make a step towards enhancing the usability and usefulness of educational design patterns using an e-learning framework approach. In engineering, frameworks have proved successful in supporting the development of complex systems by providing sources of reuse for both design and functionality. Taking this approach to the e-learning domain, we examined two different approaches of supporting the implementation of educational design patterns on existing LMS. The ‘extension’ approach is based on the provision of custom LMS components complementing the design patterns and integrated into the LMS; this approach was demonstrated using the patterns and components created in the beehive project. The other approach called ‘facade’ takes the LMS as given and provides a facade component for configuring existing LMS components to support particular educational design patterns; this approach was demonstrated by the PatMan prototype, which implements a layer on top of an LMS to provide design pattern-based configuration of components already available within the LMS.

Both approaches have their advantages and disadvantages; however we propose that future educational design pattern initiatives, particularly in the area of technology-enhanced education, should consider either one of these approaches as part of their design patterns deliverables. This would contribute to an enhanced user-centeredness and usability of educational design patterns.

A further step, which would take the approach presented in this paper to the next level, is the conception and development of a pattern-driven, extensible LMS based on an e-learning framework approach as proposed in this paper.

Acknowledgements

This research was supported by the ‘Technology Enhanced Learning’ project (SP395001) funded by the University of Vienna, and by the Australian Research Council (LP0561882). The second author also acknowledges the contributions of Dr. A. Turani to the project.

References


E-learning frameworks


