Web Services for Blended Learning Patterns

Michael Derntl, Juergen Mangler
Department of Computer Science, University of Vienna, Austria
{michael.dernl, juergen.mangler}@univie.ac.at

Abstract

Web services currently draw the attention of learning technology researchers and practitioners to leverage the degree of interoperability and extensibility of current and future learning platform solutions. One very recent research thread in blended learning concentrates on capturing Web-based learning processes and their subsequent instantiation on learning technology in the form of reusable patterns. This paper presents CEWebS, an open Web-service-based learning technology architecture designed for supporting the implementation of these patterns with conceptual guidance by the layered Blended Learning Systems Structure (BLESS) model.

1. Introduction

Though current learning platforms offer valuable tools for content management, publishing, authoring, and computer-mediated communication, there is still a sense of confinement in terms of technical interoperability and extensibility: If a learning platform solution does not offer means to adequately support a desired learning scenario, didactical course design considerations have to be traded for compromise solutions restricted by available technology. Presumably, this stems from the fact that the current breed of platforms is mostly functionality-centric and monolithic in system design. They offer learning technology atoms [5], but the composition of useful molecules is still solely up to the course/instruction designer. The functionality offered is technologically powerful and versatile, but often not targeted in terms of learner support in concrete application scenarios.

Blending classroom teaching and learning with the use of Web technologies is currently one of the major threads in learning technology research. Thereby, recent efforts concentrate on capturing successful blended learning practices [5] and design experience inherent in existing learning management systems [3] in the form of reusable design patterns [1]. Additionally, basic technical requirements and architectural considerations of e-learning systems are meanwhile quite well defined, e.g., by the IEEE/LTSA (Learning Technology Systems Architecture) draft standard [6]. However, we still perceive a lack of integrative investigation on the didactical and technological design dimensions. The Blended Learning Systems Structure (BLESS) model we introduced in [4] addresses both of these dimensions by considering their reciprocal influences: On the one hand, learning technology provides new, enhanced means of learning support, while on the other hand didactics have to be reconsidered accordingly to make situated and targeted use of learning technology.

![Figure 1. The BLESS model.](image)
we have used different learning platforms and their features to implement the patterns. But as the paper will show, no solution we used provided satisfactory support in this enterprise. Therefore, we began to build an open source, Web-service-based architecture that should be capable of supporting blended learning patterns.

The paper is structured as follows. The next section discusses general Web service issues and identifies their applicability for learning technology solutions. Section 3 presents the Cooperative Environment Web Services (CEWebS) architecture that provides a generic framework and runtime environment for e-learning Web services, and section 4 exemplifies the use of CEWebS for systematically implementing blended learning patterns. The final section presents conclusions and an outlook on required further work.

2. Web Services and Learning Technology

The distributed, heterogeneous nature of the Web calls for applications that interoperate independently of internal implementation. The Web Services Architecture (WSA) [10] provides such means of standards-based application-to-application interaction based on publicly exposed, well-defined interfaces. The four magic words (or layers) in the WSA are discovery, description, messaging, and communication. Put simply, each Web service is defined by a public Web Service Description Language (WSDL) file, which is basically an XML file that contains definitions of the operations offered by the Web service, as well as involved data types (＝description). In addition, the Web service provider may publish detailed information regarding a specific service in a Universal Description, Discovery and Integration (UDDI) directory, which eases locating Web services over the Internet in a yellow-pages-like manner (=discovery). The actual data exchange between clients and Web services relies on standardized access (=messaging) and transport protocols (=communication), e.g., SOAP over HTTP.

Recently, Web services have drawn the attention of learning technology researchers and practitioners, e.g., for decentralized, integrated support of Web-based learning processes [9], for personalization of such processes in intelligent tutoring systems through Web-service-based agents [7], for contract-based provision and discovery of distributed Reusable Learning Object (RLO) repositories [8], or for enhancing the functionality and interoperability of existing learning technology applications [2], to mention a few. All of these approaches employ Web services to increase extensibility and flexibility of existing solutions and to foster standards-driven development, dissemination, and usage of desired functionality. Summarizing, using Web services for Web-based learning purposes is all about sharing and furthering open development of complex functionality and learning content. The particular approach presented in this paper aims at implementing modular blended learning patterns through Web services that interact within an open architecture as described in the following section.

3. Cooperative Environment Web Services (CEWebS)

After multiple application cycles with trials, errors, and drawn experiences, and with basic guidance by the blended learning patterns, we are convinced that none of the currently existing platforms is fully capable of supporting thoughtfully designed blended learning scenarios in an optimally situated way. Consider the following example: Students elaborate projects and you want them to evaluate the milestone solutions of their peers online, which basically resembles a simple peer-evaluation scenario. How do you project such a scenario on a learning platform that offers, for example, chat, discussion forum, and workspaces? There is no optimal solution, but there are feasible compromises such as providing evaluations as discussion postings, uploading evaluations as text files onto the teams’ workspaces, or simply circumnavigating the learning platform by sending evaluations per e-mail. However, refusing to think solely in terms of available functionality, we push to think more in terms of desired scenarios. The Cooperative Environment Web Services (CEWebS) architecture we are proposing for layer 5 of the BLESS model, in combination with blended learning patterns on layer 3 as proceeding guidelines, in our view allows for optimal solutions.

3.1 The CEWebS Architecture

As depicted in Figure 2, the central elements of the CEWebS architecture are interfaces that control the interactions of involved components:

- **Web Services (WS).** WS’s are the central components of CEWebS. Each WS implements a self-contained learning object, whereby the term ‘learning object’ is regarded in its broadest sense here: It may refer to a simple, static Web page including some content, to interactive means of online communication (e.g., discussion forums), to any didactical process (e.g., peer-evaluation), or to any useful composition thereof. Each WS, through implementing the Delivery interface, supports a number of delivery commands that control the visual output of the WS (we will come back to this later). The Report interface defines a number of operations to generate reports (e.g., learner activity within the WS) that are output to the requesting browser using HTML or any type of binary data such as PDF, or that may send messages to learners (e.g., notification of those who have missed a deadline). Finally, the Administration interface
defines functionality needed to control the administrative parameters of a WS, e.g., involved students and teams, whereby each WS is capable of hosting multiple instances of that data. This way it can be used concurrently by more than one Transformation Engine. By implementing all of the described WS interfaces, any provider may contribute a WS that encapsulates some Web-based learning content or process.

**Transformation Engine (TE).** The TE can actually be considered as the ‘learning platform’ or, alternatively, the Web service container: It receives the connection properties (i.e., server URL, port, name, etc.) of a set of distributed WS’s, which are configured by the Administration Manager, via the Maintenance interface. The TE keeps that list of WS instances and users within its realm. Any TE is hosted on a Web server and is publicly accessible through standard HTTP requests: Upon receiving an initial (empty) request, the TE calls the first WS in its list of services over SOAP via the Delivery interface to display itself by sending its configured default command along with the unique instance identifier of that WS. The WS in turn returns a raw response (which is basically a subset of HTML) that represents its contents for the requested command. The TE transforms the WS response and constructs a complete HTML page including page headers, footers, and other “surrounding” elements that make up a user-friendly Web application. That HTML page is returned to the requesting Web client, which is typically a standard Web browser used by a student to access the learning platform.

**Administration Manager (AMan).** To configure the TE’s and WS’s, administrators are provided with an AMan, which is basically a Web application that interacts with WS’s through the Administration and Notification interfaces as well as with TE’s through their Maintenance interfaces. The core administrative task is to configure each TE along with its WS’s, which represents the learning platform for a particular course. Additionally, the Notification interface empowers the AMan to keep shared data (e.g., users) consistent among a TE’s Web services.

Currently, our initial set of Web services has to be configured more or less manually by editing their XML configuration files. However, to be able to address a broader, technically less experienced audience, we are currently in the inception phase of a wizard-based WS configuration application, leading the administrator through any required step in the WS configuration process. Thereby, it will be required for each WS to provide an XML Schema-based meta-description of its configuration capabilities. In this way, its custom administration wizard can be created automatically from the meta-description of the WS.

**Report Manager (ReMan).** The ReMan, which resides within the same Web application as the AMan, is used by instructors to retrieve reports through the WS’s Report interfaces (compare the WS description above). Actually, supporting the instructor in generating meaningful reports of students’ activities was initially one of the driving factors for designing CEWebS, as none of the learning platforms we used so far was able to produce such reports. For example, when assessing a grade for a student’s contribution, the ReMan assists the instructor in generating peer-evaluation reports for this contribution.

We have already completed a reference implementation of the CEWebS architecture and used it in several courses during the previous and current academic year. Thereby, the TE was implemented in PHP, some of the WS’s were implemented in Smalltalk and some in C# .NET. We have imposed the usage of different technologies and the distribution of components among several server computers on ourselves to substantiate the interoperable nature of CEWebS as well as to keep an eye.
on performance issues to deal with in networked applications.

3.2 Advantages and Constraints

Using the CEWebS architecture has shown to produce some major advantages over standard learning platforms:

(1) Through employing standard communication protocols the architecture is designed for location transparency of Web services and involved components, i.e., it does not matter on which computer the components are hosted. They may all be run on a single server, but may as well be distributed all over the Internet. Components may be implemented by different providers using heterogeneous software and hardware platforms.

(2) The architecture may be incorporated into any existing learning or content management platform by implementing (at least) the TE. Thereby, the host platform user interface layout can remain unchanged – only the actual content provided by the Web services has to be transformed to visually fit the host platform by doing appropriate transformations (e.g., XSL, CSS, etc.) on the WS’s raw responses.

(3) One convenient feature of CEWebS is that administration is only due where it is really required. As opposed to many closed, monolithic platforms, one does not have to excessively assign user rights to a WS that, for example, is just provided for displaying public, static information. The authentication mechanism that CEWebS employs is very simple: Before sending any display command to the Delivery interface of a WS, the TE asks the WS whether it requires user authentication for that command.

Besides its benefits, the design of CEWebS entails some technical constraints that mainly concern integration among different WS’s: It is, for example, currently impossible to compose one single Web page from the output of multiple WS’s. After extensive discussions we made the decision to not support this at the current stage of elaboration. The main reasons were: (a) adhering to the guiding principle of simplicity, (b) keeping TE’s free from complex logic, and (c) keeping the Web service communication overhead at a level that remains imperceptible to end users. Nevertheless, each WS may include references to other WS’s (which are transformed into proper hyperlinks by the TE) in its output.

In addition, the architecture consumes significantly more network communication and computation resources when compared to other popular server-side environments: Each page rendering cycle includes two request/response Web service invocations (prepare and do), each including marshalling of various complex parameters, as well as subsequent Web page composition via output transformations performed by the TE.

Finally, when its distributed nature is exploited by employing Web services from multiple dislocated providers, it becomes increasingly complex to manage the components used in one configuration, especially in terms of troubleshooting and performance tuning.

4. Example of Use

As an example, Figure 3 shows the bottom three layers of the BLESS model for the Reaction Sheets pattern, which is located in the Feedback package (see layer 3). It describes a scenario where students provide written feedback to the instructor regarding specific learning activities, which is one important collateral element in our teaching activities. The Web templates on layer 4 generically describe a set of interactive Web pages capable of putting the reaction sheets pattern into practice:

(1) A start page for general information on the reaction sheets, for links to the reaction sheet submission form, and for links to previously submitted sheets that are publicly accessible to all students (as opposed to private sheets that can only be reviewed by the instructor using the Report Manager).

(2) A page for viewing public reaction sheets that have already been submitted by students.

Figure 3. Instantiation of the Reaction Sheets pattern on CEWebS.
(3) A page that provides a Web form for submitting a reaction sheet. Consequently, the Reaction Sheets Web service supports three display commands to meet the above requirements: The start command (i.e., the default command sent initially by the TE) implements the start page, the view command (whose Web page is left out in the figure) implements the page for viewing submitted reaction sheets, and the compose command implements a Web form for reaction sheet composition and submission, whereby the latter two commands require user authentication.

Even though the example is quite simplified, it shows that CEWebS provides an architecture that, optimally in combination with blended learning patterns, is capable of leveraging modular and targeted implementation of blended learning scenarios according to users’ needs.

Currently, we are using CEWebS for supporting a major course on Web Engineering including 9 lab groups with a total of about 190 students. We are applying Web services for different kinds of individual and team-based learning and collaboration scenarios such as project-based learning in teams, online workspaces, discussion forums, and elaboration of contributions based on online contracts that will be subject to peer-evaluation by all participants. Additionally, we are going to conduct qualitative and quantitative analysis of data collected by questionnaire Web services both at the beginning and at the end of the course, and by soliciting open feedback (online reaction sheets) from students on any aspect of the course.

5. Conclusion

We have presented an open, standards-based learning technology architecture that uses distributed Web services to support a broad variety of blended learning scenarios and administrative tasks. In combination with blended learning patterns, the technological dimension is complemented with didactically sound course design based on the Blended Learning Systems Structure model. In this way, both worlds are joined at their most crucial point: modular and interactive Web services for implementing modular and interactive blended learning patterns.

Future work will in the first place concentrate on improving the visible end-user components of CEWebS (mainly the Administration Website) with respect to Web usability issues, and on collecting further experiences with the architecture in upcoming courses. Before releasing an initial version of the architecture and its reference implementation along with a set of basic Web services to the public we aspire to perform some additional improvement iterations of components and interfaces to fortify broad applicability and reusability of CEWebS as well as to offer a solid starting point for learning Web service providers and users.

6. References