INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen
Oettingenstraße 67, D–80538 München

Ludwig
Maximilians
Universität
München
LMU

Web Services for Teaching: A Case Study

François Bry, Norbert Eisinger, Georg Schneemayer

appeared in *Proc. First International Conference on Web Services*,
ICWS'03, Las Vegas, USA, 23-26 June 2003.
http://www.pms.informatik.uni-muenchen.de/publikationen
Forschungsbericht/Research Report PMS-FB-2003-4, April 2003

Web Services for Teaching: A Case Study

François Bry Norbert Eisinger Georg Schneemayer
Institut für Informatik, Universität München,
Oettingenstr. 67, D-80538 München, Germany

Abstract *The area of software applications for teaching (teaching applications for short), whether for distance teaching or for traditional presence teaching, is characterized by a multitude of different requirements and expectations. A corresponding multitude of different teaching applications have been and continue to be implemented, most of them sharing parts of their functionalities that tend to be re-implemented over and over again. The paper argues that a* compositional approach *based on* web services *would overcome this deficiency quite naturally. The paper classifies the services that need to be implemented by teaching applications and proposes an* open architecture *based on web services for information interchange, integration, and interoperability among existing teaching applications.*

Keywords: contextual web services, web-based teaching

1 Introduction

The need for computer support is obvious in teaching scenarii where students interact with software components rather than with human teachers. Somewhat less obvious is the fact that traditional classroom teaching in universities can be complemented by supportive computer services, too, which benefit teaching staff or students or both. Some of those services may focus on a more efficient management of courses and a better analysis of their success. Other services may serve as catalysts for the communication between or within the different participating groups. Some services may help students develop learning skills by providing information and support adapted to their individual needs.

The authors of this paper were confronted with the question of computer support for an undergraduate course in 2001, when the number of students had dramatically increased (more than tripled) compared to previous years. The only chance to cope was to develop administrative software support [1] for registration to tutorial groups, for homework distribution, submission, assessment, and bookkeeping, and for managing the consequences all of this had on the admission to the exams.

Later courses used different regulations about homework, exam admission, and other details, raising the question of a more generic approach to this kind of teaching support. Therefore an investigation [2] was started to provide a survey and classification of existing teaching services and to work out a common framework. This paper reports on this classification and framework.

2 Existing Teaching Applications

Existing teaching applications come under three major categories: (1) Adaptive and Intelligent Web-based Educational Systems (e.g., ActiveMath [3], ELM-ART [4], WebDL [5]), which offer intelligent and adaptive support to the learner based on approaches from adaptive hypermedia, artificial intelligence and adult learning theories. (2) Course Management Systems (e.g., WebCT and Blackboard [6] AulaNet [7], OKI [8], CampusSource [9]), which assist teaching staff managing courses on-line by providing tools for student management, content authoring and management, exercises, progress tracking, communication and collaboration. (3) Assessment Systems (e.g., Web-

Assign [10] Praktomat [11]), which support exercise sessions appended to some courses by tools for electronic homework distribution and submission, semi-automatic correction and tests, peer reviews etc.

Such systems cover different aspects of teaching, yet make use of similar information and share some functionalities. Thus, their integration would be highly desirable. However, this is difficult, because many of them are isolated and/or closed systems confined to special programming environments and/or depending on proprietary data formats. An architecture enabling an integration therefore needs to be an open one.

3 A Tentative Classification of Teaching Services

Before proposing an architecture capable of integrating different services, this section classifies the services needed in the scenario. Similar endeavors often rely on *usage centric classification* schemes [12, 13, 14]. They are, however, not always adequate, because in the majority of cases the different user groups (like faculty staff, students and system support staff) work together, albeit from different directions. They access the same services, only the presentation and their level of access are different.

With this in mind, a *service centric classification* is used in this section. It is worth to mention that most of the services presented here are not specific to teaching. For example, most of them are relevant to cooperative work in general, with a small difference: there is no such clear distinction of the roles in teams as there is between students and staff. In cooperative work there is only one team, perhaps organized by some hierarchy.

Teaching Material. Services that enable content storage and content presentation.

The content storage services support the integration of different legacy formats. It should be possible to enhance the legacy content with meta-data for the use with other services.

The content presentation services implement access control functionalities. In addition, meta-data enriched content can be presented in various forms depending on the users' needs and contexts.

Individual and Group Browsing. Browsing services, such as the suggestion of related information or an advanced history functionality that is aware of changes and updates to previously accessed content, are mainly based on analyzing the navigation behavior of users.

More advanced services are possible if explicit interaction with the user is possible, e.g. with the help of rating or private annotation systems.

Communication. To be useful, asynchronous communication systems need a lower user base than synchronous communication systems. Such a lower user base is very likely in the scenario of teaching services as support for traditional teaching. Because of their asynchronous nature, services like discussion forums, public annotations, user moderated pages or wikis, and announcements are the most useful communication services in this scenario.

Homework Management and Assessment. Services in this area are characterized by schedules, prerequisites in form of previous work, etc. This shows that although services for homework management and assessment seems to be very specific, variants of such services are applicable in the domain of team work, too.

Examples for services in this area are parameterizable on-line accessible exercises, online and incremental submissions of work, services enabling peer review, and automated testing before and after submission.

Progress and End-of-Course Statistics. Every interaction with the system creates data that can form the basis for various statistics. These statistics can help to improve the system by analyzing how the users actually use it. But also the students can immediately

benefit from various statistics, like statistics about their performance in comparison with the average student. Such informations can be used to alert both lecturers and individual students almost real-time when a drift from the expectations occurs.

Course Management. Course management services support both students and lecturers in administrative tasks. This characterization would also apply to some of the services described above. But course management services are meant to be for purely administrative tasks. Examples are registration, admission and document generation services. These are all very basic services, and therefore it is very likely that implementations of these services already exist as part of other services. In an integrated system these different implementations need to be merged in some way.

4 Towards an Open Architecture for Composable Teaching Services

This section proposes an integrative approach based on web services, starting with a summary of the goals such an approach ought to meet.

Open for Existing Management Infrastructure. University courses are embedded in an administrative environment that uses its own computer support. Part of this environment may be an enrollment office keeping student records with data that is also needed for the organization of a course. Another part of the environment may be the local computing center keeping account data that can be used for authentication to grant students access to individualized services. Whatever systems are being used in the administrative environment, it should be possible to make accessible selected parts of their data while ensuring privacy and security.

Open for Existing Application Infrastructure. Existing application infrastructure includes services that may be required by any kind of software, e.g. for data storage, data exchange, logging, or security, but also more specialized services that may be required by web software, e.g. for communication with the user's browser, more reliable security, authentication, or authorization. The use of such existing application infrastructure should be possible in order to give a head start to the project.

Open for Existing Standards. Standards are primarily a means to achieve interoperability. Many standards relevant to web based teaching services exist or are being developed, e.g. LOM [15] for the exchange of educational content, SOAP [16] for communication between services, Media Queries in CSS3 [17] for adaptive rendering of content. The use of any such standards should be supported by the system architecture.

Composable with Existing Teaching Applications. Adaptive and intelligent web-based educational systems, course management systems, assessment systems, as well as other teaching applications are already in use today. There may be reasons to continue using them, e.g. because of organizational requirements, or just as a convenience for staff used to them. No matter what the reasons are, the integration of existing teaching applications should be possible.

Open for the Use with Existing Teaching Material. Lecture notes, handouts, slides, and similar material to be distributed to students exists in many different formats. If a new integrative approach required its own proprietary format or even some given standard format, it would have to start up lacking exactly the kind of material that abounds in other formats. It should be possible to make use of existing teaching material, either through a one-time import mechanism or through an adapter for continuous integration.

Open for Different Needs. Administrative regulations, technical reasons, the number of par-

ticipants and similar factors influence the possible organizational form of a course. For example, if students have to complete two or three medium-sized programming projects during a semester, it may make sense to organize some kind of peer reviews among them. If students have to submit homework every week, peer reviews might still be possible in principle, but hardly feasible in practice. And if students have to take regular written tests, peer reviews are probably not even permitted. Thus, whether or not a "peer review component" is needed depends very much on the special circumstances of the course. The same can be said for almost any need of a course. It should be possible to meet special needs of courses as they arise without anticipating all possible needs.

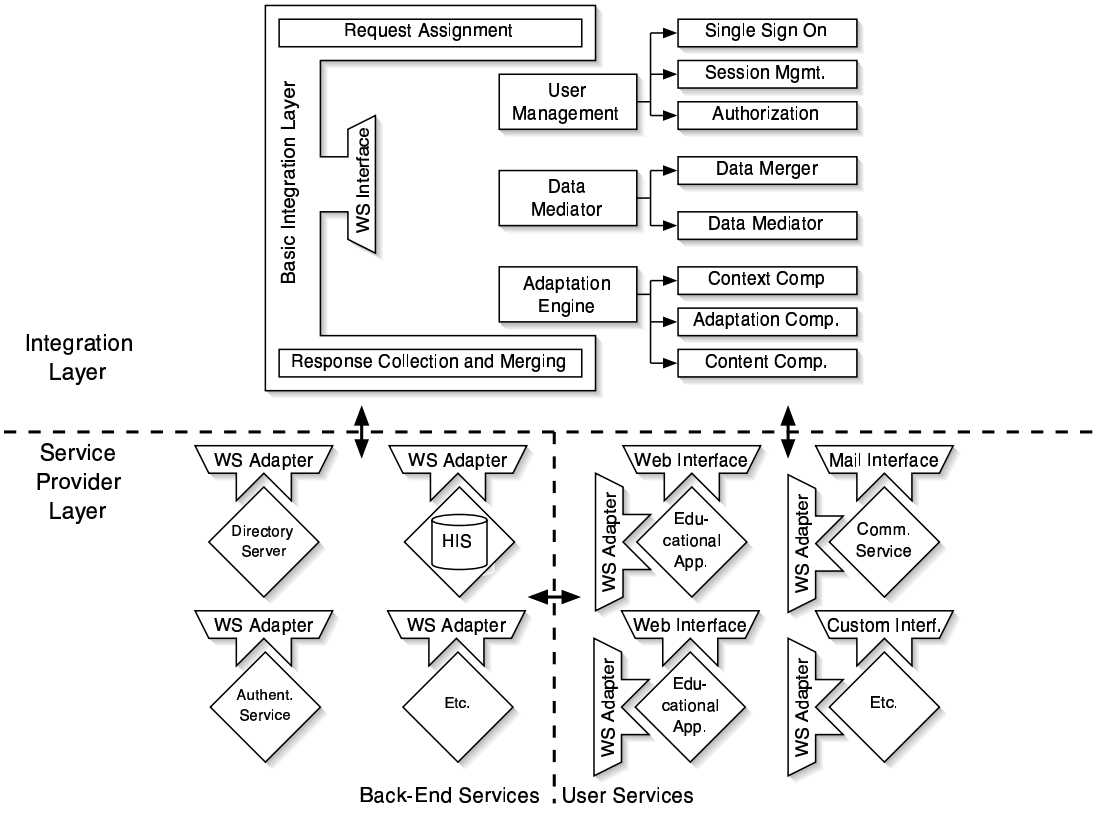

Figure 1: Architecture for Open and Composable Teaching Services

Figure 1 shows the proposed open architecture to meet the goals listed above. It consists of two major layers: the integration layer (a kind of service request layer) and the service provider layer.

The integration layer is responsible for the interaction with the user. It processes requests from the user and builds the response. It assigns the service providers to process the user's request, collects their individual responses, and combines them into an integrated response.

Service providers are equipped with web service adapters in order to allow their integration with

other service providers and with the integration layer. One group of service providers consists of back-end services, which are mainly existing management infrastructure used by other services or by the integration layer, but not of direct interest to the end user. The other group consists of services with a user interface, which are existing educational applications or auxiliary applications or services.

4.1 The Integration Layer

A light-weight integration layer could be implemented as a so-called Enterprise Information Portal, which simply combines the user interfaces of the individual service providers. Existing frameworks or server applications such as Apache Jetspeed [18] make such an implementation possible with relatively little effort, thus enabling an early start of an operational system. The somewhat limited benefit of this light-weight implementation can then be scaled up by adding more advanced components like the following:

A *user management component* would start with a uniform sign-on service that handles authentication with the individual service providers. The next step would be a full session management. If the integration layer is aware of user classifications such as student, teaching assistant, lecturer, the user management component can provide an authorization service allowing or denying access to integrated views of underlying service providers.

A *data mediator component* is useful if different service providers use data with overlapping content, but cannot be configured to share a common data storage. Then the mediator component can trigger the relevant updates to ensure consistency of the different data storages, or it can organize more complex schemes of data merges.

An *adaptation engine* does not affect adaptive features the integrated applications already have. Rather, it provides additional adaptation, for example to different capabilities of the client device. It can also adapt the combination of different service providers to the user's present context: a communication service may be integrated if the context is "exploration", but not if it is "examination".

4.2 The Service Provider Layer

The W3C's Web service architecture working group [19] defines web services as follows:

> A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.

Service providers in the proposed architecture are existing applications or systems, which are not changed but extended with web service adapters. The adapters translate between the legacy interfaces and the text based XML messaging, thus providing the required web service functionality independent of implementation languages, platforms, locations etc. This very loose coupling allows the replacement or upgrade of single services without affecting other services using them.

4.3 Discussion of the Proposed Architecture

Teaching is not the only domain requiring common web interfaces to existing applications and information sources. A company with its grown infrastructure of electronic applications and its needs to make this infrastructure accessible over the company's intra net faces very similar problems. Solutions to such needs are called Enterprise Application Integration servers and Enterprise Information Portals. The proposed architecture was inspired by these solutions.

A thorough analysis [2] with respect to the goals listed above shows that the proposed architecture meets all goals, whereas other proposals meet only some of them. Moreover, the approach enables an easy start with a thin integration layer combining few existing applications with little add-on functionality, which can be smoothly upgraded by adding more service providers and/or more integration layer functionality, while the whole system remains operational at each extension step.

A possible drawback is that web services are a new approach and no mature technology. Specifications (e.g., SOAP [16]) are evolving very fast, and their implementations (e.g., Apache Axis [20]) for the different programming environments are updated very frequently.

In addition to the architecture an implementation would also require the selection of appropriate web languages for the specification of and reasoning with ontologies, such as OWL [21]. Before that step the conceptual work needs to be done.

References

[1] Georg Schneemayer, "Einsatz elektronischer Medien für den Übungsbetrieb," Project Thesis, Institut für Informatik, Universität München, 2001, http://www.pms.informatik.uni-muenchen.de/publikationen/#PA_Georg.Schneemayer.

[2] Georg Schneemayer, "Contextual Web Services for Teaching," M.S. thesis, Institut für Informatik, Universität München, 2002, http://www.pms.informatik.uni-muenchen.de/publikationen/#DA_Georg.Schneemayer.

[3] Erica Melis, Eric Andres, Jochen Büdenbender, Adrian Frischauf, George Goguadze, Paul Libbrecht, Martin Pollet, and Carsten Ullrich, "ActiveMath: A Generic and Adaptive Web-Based Learning Environment," *Artificial Intelligence in Education*, vol. 12, no. 4, 2001, http://www.ags.uni-sb.de/%7Eilo/articles/GenericAndAdaptiveWebBasedLear%ningEnvironment.pdf.

[4] Gerhard Weber and Peter Brusilovsky, "ELM-ART: An Adaptive Versatile System for Web-based Instruction," *International Journal of Artificial Intelligence in Education*, , no. 12, 2001, http://www2.sis.pitt.edu/~peterb/papers/JAIEDFinal.pdf.

[5] Jesus G. Boticario and Elena Gaudioso, "A Multiagent Architecture for a Web-Based Adaptive Educational System," Tech. Rep. TR SS-00-01, 2000, http://www.ia.uned.es/~jgb/publica/download/aaai00-s.ps.

[6] Sabine Siekmann, "Which Web Course Management System is Right for Me? A Comparision of WebCT3.1 and Blackboard 5.0," 2002, http://astro.temple.edu/~jburston/CALICO/review/webct-bb.htm.

[7] Hugo Fuks, Marco Aurelio Gerosa, and Carlos Jose Pereira de Lucena, "The Development and Application of Distance Learning Courses on the Internet," *The Journal of Open and Distance Learning*, vol. 17, no. 1, 2002, http://139.82.35.97/groupware/publicacoes/openlearning.pdf.

[8] Jeff Merriman, "What is OKI?," 2002, http://web.mit.edu/oki/product/whtpapers/whatis.html.

[9] H.-W. Six, J. Voss, and W. Schäfer, "Architekturschema für VU-Systeme," 2002, http://www.campussource.de/projekte/architektur/architektur_vusysteme.html.

[10] H.-W. Six and Frank Albers, "WebAssign - General Information," 2002, http://niobe.fernuni-hagen.de/WebAssign/index_eng/index_eng.html.

[11] Andreas Zeller, "Making Students Read and Review Code," 2000, pp. 89–92, http://www.infosun.fmi.uni-passau.de/st/papers/iticse2000/iticse2000.pdf.

[12] Bruce Landon, "Online Educational Delivery Applications: A Web Tool for Comparative Analysis," 2002, http://www.c2t2.ca/landonline/.

[13] Sabine Siekmann, "Which Web Course Management System is Right for Me? A Comparision of WebCT3.1 and Blackboard 5.0," 2002, http://astro.temple.edu/~jburston/CALICO/review/webct-bb.htm.

[14] Edutech, "Comparision of Web Based Course Environments," 2002, http://www.edutech.ch/edutech/tools/comparision_e.asp.

[15] IEEE, "Learning Object Metadata," 2002, http://ltsc.ieee.org/doc/wg12/LOM_WD6-1_1_without_tracking.pdf.

[16] W3C and Nilo Mitra, "SOAP Version 1.2 Part 0: Primer," Jun 2002, http://www.w3.org/TR/2002/WD-soap12-part0-20020626/.

[17] W3C, "Media Queries," Jul 2002, http://www.w3.org/TR/css3-mediaqueries.

[18] Apache Software Foundation, "Apache Jetspeed Enterprise Information Portal," 2002, http://jakarta.apache.org/jetspeed.

[19] W3C, "W3C Web Services Architecture Working Group," 2002, http://www.w3.org/2002/ws/arch/.

[20] Apache Software Foundation, "Apache Axis," 2002, http://xml.apache.org/axis.

[21] W3C, "OWL Web Ontology Language 1.0 Reference," Jul 2002, http://www.w3.org/TR/owl-ref/.