The Service Architecture in the ACTIVEMATH Learning Environment

Paul Libbrecht, Stefan Winterstein
DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken
Germany
paul@activemath.org

Abstract

We discuss the usage of web-services in the ACTIVEMATH *learning environment, describe the prototypes implemented and the experience gained.*
Requirements for an ideal web-service technology are described. They include the notion of event propagation as well as the consideration for stateful web-services. Finally we propose models to make the web-services of ACTIVEMATH *available to client components without affecting privacy or security.*
Further work, within the LE ACTIVEMATH *EU project is described.*

Introduction

A service oriented architecture provides a conceptually easy paradigm to encapsulate functionalities of common usage. The Web-services recent technology boom proves it.

In this article, we wish to present the service architecture of ACTIVEMATH as it is currently implemented and the service architecture being developed within the LE ACTIVEMATH EU project.

After a short presentation of ACTIVEMATH and its web-services, we formulate ideal qualities that a web-service technology could have provided us. We continue with a description of methods to let client components access the services and conclude with the experience gained with the implemented protocols.

1. About ACTIVEMATH

ACTIVEMATH is a web-based learning environment. It presents mathematical learning content in a web-browser, and supports this presentation with more active learning parts such as interactive exercises. The presentation is the result of a transformation process from a semantically encoded XML content stored in a database. ACTIVEMATH maintains a learner-model depicting the estimated learners' knowledge, it updates it by tracking reading actions and using the results of interactive exercises.

More about the system can be read in [7] or from the project's web-page, `http://www.activemath.org/`.

2. Web-services in ACTIVEMATH

Since its early start, in 2001, ACTIVEMATH was designed to use three major services:

- the content-database, commonly called *MBase*, whose role is to store the XML files, serving fragments by ID, and resolving their relations
- the learner-model whose role is to store tables of estimated knowledge and update it following user actions
- the mathematical systems' service, responsible to manage the lifecycle of sessions with computer algebra systems.

The three services can be living on a remote process and are accessible through an XML-RPC (see [2]) interface.

Web-services provide an abstract interface which can be used by different tools. Deploying these *components* of ACTIVEMATH as web-services has eased up considerably the development, allowing, for example, a developer to only run the core server, or only a service.

Based on this abstraction, various deployment schemes could be implemented, such as multiple web-servers using a single user-model, or distributed content-databases. What is more important, this abstraction has allowed differently scaled applications to access the same server using the same interfaces and client classes: for example, the prototype table-of-contents client component (see later) uses the same content database interface as the presentation process.

3. Web-service technologies for ACTIVEMATH

To connect these services, the XML-RPC-protocol was chosen as a simple, efficient, and widely available standard. At the time, this choice was mostly dictated by the available implementations and the simplicity of the standard as an implementation in LISP language and on the Mozart-Oz platform was to be done as well. SOAP has, since then, imposed its predominance and has made possible a large amount of extensions.

For the web-services in ACTIVEMATH to be ideally developed, a number of requirements should be satisfied:

1. as in all web-services, we expect a web-service protocol to be an extension of the HTTP protocol and be using XML as content encoding.
2. A web-service interface is a specification which has to be published for human consumption (as a documentation) and for machine consumption (for example in a development environment).
3. similarly, the multi-step processes that several web-services take part to should be specifiable in both human and machine form.

The SOAP standard clearly answers the first requirement. Although the WSDL standard [1] allows the specification of the interfaces and is well supported in development environments, it seems to have no support, or have been little used, to produce a human readable description of the web-service which other developers could be reading. Similarly BPEL4WS or WS-Choregraphy languages care for the third requirement, just as OWL-S, but none of these standards seem to care for human documentation.

For the lack of complete fullfillment of these requirements, and so as to take advantage of the lightest components, ACTIVEMATH has not migrated from SOAP yet.

4. More Web-services Paradigms

The "remote procedure call" paradigm provided by classical web-services is not always appropriate. We propose two important paradigms which, we believe, are needed for a better usage of web-services in ACTIVEMATH:

4.1. Stateful Web-Services

Some principles of web-services state that a web-service should be stateless. We believe this is unusable in many situations.

The consideration of a web-service as a stateful object is, for example, a requirement for the web-service of a session in a computer-algebra system for an interactive exercise as described in [5]. It is also a requirements when objects are hard to serialize without information loss.

The presentation of Steven Linton [6] shows that references to objects like mathematical groups within the GAP computer-algebra-system make much more sense as the serialized version: any serialization of a group is bound to loose information which would need a possibly heavy recomputation.

The foundational work on representational state transfer services in [3] has offered practical approaches to consider resources as states and has launched the REST architectural style. The REST architectual style promotes resources as building blocks representing the state of a resource and proposes simple usages of the HTTP methods to perform read, act, and write operations.

We hope to see the WSRF [14] standard gain a wider audience, not forgetting the simplicity principles promoted by the original REST description as described, also, in [12].

4.2. Event Model

When considering the interface description of a learner-model web-service,[1] it becomes apparent that many functional calls are mere *information-messages* as opposed to imperative function-call. Such notifications are better formulated as events:

In the common terminology, a component can choose to *publish* events, making it an *event source*. Another component can *subscribe* to the events published by an event source, making it a *listener* (or event *consumer*), which will then receive *event messages* from the event source (sometimes also called *notifications*).

In contrast to a full-fledged messaging model, events are not sent from a specific sender to a specific recipient, but rather remain anonymous: when creating and publishing an event, the component is usually not aware who is listening to the events (only the module managing the subscriptions is). Also, the listener typically does not care which component or module created the event, it only knows where to subscribe to the events it is interested in.

The delivery model of events is typically limited to an asynchronous *push*, meaning that events are delivered from the source to the listener without waiting for the listener to react to the notification or return a result for it.

The event model fits well in the architecture of ACTIVEMATH.

Most the learner-tracking information, which includes information such as the reading time of a content-item or the score of an interactive exercise session, is best formulated as events.

A typical event propagation example could be the following: using a *software eye-tracker* (see [15]), one can

1 The user-model XML-RPC API can be interpreted from the Javadoc of its XML-RPC server: `http://www.activemath.org/~ilo/redirs/XmlRpcUserModel.html`

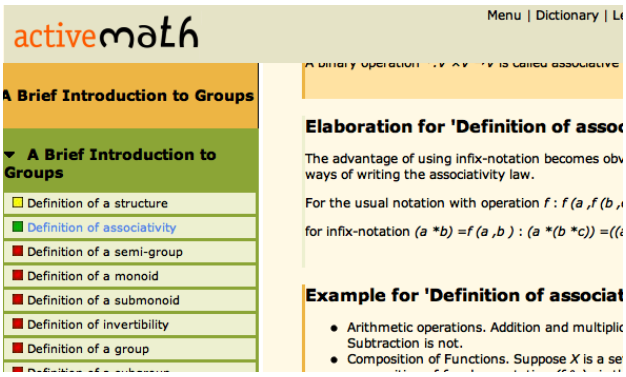

Figure 1: Extract of a book view in ACTIVEMATH.

measure the time taken for a learner to read each paragraph. This time needs to be reported on the server some point later. Arriving in the servlets responsible for content-browsing, the event needs to be notified, at least, to the learner-history who should write this down. A follow-up of this event should be produced for learner-model updates (indicating a higher probability of rote knowledge of related concepts). Some time later, the learner-model will probably fire an event that the concept's knowledge value has been changed. This information flows back until the client-browser: the tables of contents of books are presented with a colored bullet aside of each page indicating the estimated mastery value for the concepts of this page as pictured in figure 1.

5. Client Access to ACTIVEMATH Web-Services

When planning a web-server development, a common requirement is to have a rich user-interaction on the client. Depending on the type of interaction, the plain browser technologies are often not enough and code to run in client user-interfaces are needed.

Client code-components can take advantage of using the web-services available in the server. This is especially true if it is possible to share the code used on the client and the server like the Java platform allows.

In the LE ACTIVEMATH-project[2], two *rich client* user-interfaces are planned:

- the assembling-tool should serve as an easy to use storage for reference to items seen in ACTIVEMATH and outside. It should provide a tree view of the content organization which should be easily modified.
- the concept-mapping tool should offer graphical presentations of the relations between the knowledge items and allow such a presentation to be interactively modified.

The content-database is of use for both of them to attach a type (hence an icon) and a title to the ID of each element or to extract content.

The learner-model can be of use for the assembling-tool, wishing to support the learner by information related to items she has been browsing recently or exercises she has been performing recently. Moreover, events have to be sent from these components (for example notifications of exercise start and end) and be received.

Doing so, however, cannot be done freely as it would open such information as the learner-model content to the world. A form of authentication is needed to ensure that only this client accesses the server. The privacy issues related to the storage of learner-history and learner-model diagnose has been recognized as a delicate legal case in [4]. It should thus, be provably impossible to access the learner-model in an inappropriate way. Moreover, the code running on the client may be abused, and act malevolently under the effect of hackers or virusses. It appears thus reasonable to bring the risk to its minimum and remove the risk of accessing the learner-model of another person by client components.

A last requirement to offer these web-services to client is the tolerance to strong firewalls leaving only proxied-HTTP requests on the default port number come out of their network. This requirement forces the same port (hence the same server) to be used by the web-service and the content-server.

5.1. Proposed implementation

To answer these requirements, we shall be building web-service relays. They will be created dynamically at a random URL within the web-server upon authenticated request. Moreover, these relays will be aware of the web-service interface semantic to filter their requests according to such criteria as the restriction to a user.

Event propagation will happen using extra web-service interfaces; for events reception, the client components will need a polling channel since downstream connections are impossible.

By the addition of such relays we believe having solved the delicate goal offering web-services to client components without changing the code of the web-service server or client interfaces.

6. Prototypes implemented

The ACTIVEMATH system described here is implemented and available. The content-database and learner-model services are implemented. Together, they serve well

2 See `http://www.leactivemath.org/`

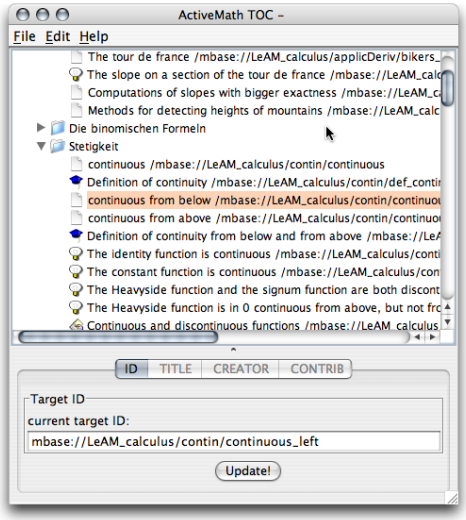

Figure 2: The table-of-contents editor.

a classroom of 30 students with the content of a book, including providing intelligent planning of learning content.

6.1. Interactive Exercises with Computer Algebra Systems

Interactive exercises wrapped as sessions in a computer-algebra-system have been implemented [5]. They present a terminal-like interface where learners provide steps of exercises which gets evaluated by the computer-algebra-system and by the (authored) exercise-evaluator. The computer-algebra-systems used were MAPLE and MUPAD, they are wrapped by the MATHWEB software-bus [16] and exposed as XML-RPC web-services. The applet used for this interaction is using an XML-RPC connection to a *proxy* on the server which, itself sends the computations to the computer-algebra-system and returns the feedback.

Both the web-service of the proxy and the web-service of the computer-algebra-systems are stateful services and they are, actually, created on the fly by a factory service.

The experience gained with this interactive session settings has shown that the management of the lifecycle of-sessions should not be underestimated: leaving unused computer-algebra-system sessions unterminated was experienced to cost a lot of CPU whereas obtaining clean end-of-life signals of an applet on a server is often impossible. To this effect, the terminal console signals its life to the proxy every 3 seconds and lack of these signals after 10 seconds releases the computer-algebra-system session.

6.2. The Table of Contents Editor

The table-of-contents editor is pictured in figure 2. It was originally conceived as a tool for an author to create books by assembling a set of chapters and pages. We have made possible to to open this editor as a learner in ACTIVEMATH so as to edit the content of books (*assemble a book*) with content units fetched in other parts of ACTIVEMATH. The table-of-contents editor is not, anymore, inside ACTIVEMATH and is being replaced by the assembly-tool.

The table-of-contents editor was started using a Java Network Launch Protocol [13] application descriptor generated on the fly and downloaded by the browser. This allowed the server to receive an authenticated browser request and relay the authentication as a *ticket* inside the application descriptor. Using this ticket, authenticated requests could be performed from the editor to read and write the table-of-contents. The table-of-contents editor is using the content-database to indicate the type of items manipulated. It does its connection directly, expecting a *default* deployment of the content-database on the server, and no firewall under way...

The table-of-contents editor stumbled also accross the lack of event architecture: when saving a table-of-contents, the storage on the server was updated, however, there was no events fired to notify the change in this storage which would, then, reload the necessary parts of the book display, the learners had to expressedly request the reload.

6.3. Further Research Within the LE ACTIVEMATH EU Project

The LE ACTIVEMATH EU project is an FP6 between several Universities. We describe here the research that is being pursued in this project relevant to our architecture:

The event framework has been implemented in-virtual-machine and the remote event distribution is under work. The target of a remote web-service as event publisher and listeners is to be clearly separated from the targets of a client java component or a client javascript running in a browser

A revised learner-model is being worked on by the University of Northumbria in Newcastle with a strict reliance on the information flow of events. New values will be maintained such as the autonomy and approval measures [11] and further user-actions tracking will be used such as digests of low-level mouse-movements.

The assembling tool will replace the table-of-contents editor. Its functions will include a broader learner-support, as *extended-memory* of the learner, somewhat similar to

a bookmarks window. Pointing to ACTIVEMATH content units, and managing these references will still be possible and we intend to implement the service-relays described in section 5 in order to access, at least, the content-database and book-storage.

The Concept Mapping Tool is implemented [8] but needs to be further integrated. It will be used, among others, in exercise situations, requesting to a learner to organize concepts, comparing, then, the result with the content database. The content-database service will, thus, be used. Moreover, events will be sent by the concept mapping tool to notify start and ends of exercises as does the exercise system on the server-sides.

7. Related Work

Usage of web-services is not new in the e-learning community. We have attempted to provide complete practical solutions which could be deployed in todays environments.

The usage of secured and authenticated web-services is very young, a quick overview is in [10]. The solution we have proposed seems to be simpler and its requirements are known to be running in applets.

Conclusion

In this article we have presented the usage of web-service technologies for the development of the ACTIVEMATH learning environment. We have shown the difficulties encountered and approached solutions which are under work within the LE ACTIVEMATH EU project.

Web-service technologies are in a boom. Based on the SOAP specifications, a myriad of *modular* specifications arise. Combining or using these technologies, even though everything was planned to be compatible, tends to be hard and experiences tend to be precious. A proof of this statement can be found in the creation of a technical comittee in the OASIS organization [9].

We have attempted to provide simple and practical answers which have brought us the advantages of web-services in the development of the learning environment while respecting user-privacy.

References

[1] E. Christensen (Microsoft), F. Curbera (IBM Research), G. Meredith (Microsoft), and S. Weerawarana (IBM Research). Web services description language (wsdl) 1.1, March 2001. See `http://www.w3.org/TR/wsdl`.

[2] Dave Winer. XML-RPC specification, October 1999. `http://www.xmlrpc.org/spec`.

[3] R. T. Fielding. Representational state transfer (rest), chapter 5 of "architectural styles and the design of network-based software architectures, doctoral dissertation, university of california, irvine", 2000. Available at `http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm`.

[4] A. Kobsa. Personalized hypermedia and international privacy. *Communications of the ACM*, 5(45):64–67, 2002.

[5] P. Libbrecht, A. Frischauf, E. Melis, M. Pollet, and C. Ullrich. Integration of mathematical systems into the activemath learning environment. In P. Wang, N. Kajler, and A. Diaz, editors, *ISSAC-2001 Workshop on Internet Accessible Mathematical Computation*, 2001. `http://icm.mcs.kent.edu/research/iamc01proceedings.html`.

[6] S. Linton. Interfacing with gap. Talk presented in the Mathematics on the Semantic Web Workshop (`http://www.openmath.org/meetings/eindhoven2003/index.html`), Eindhoven, May 12-14 2003.

[7] E. Melis, E. Andrès, J. Büdenbender, A. Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. Activemath: A generic and adaptive web-based learning environment. *International Journal of Artificial Intelligence in Education*, 12(4):385–407, 2001.

[8] J. Müller, M. Mühlenbrock, and N. Pinkwart. Towards using concept mapping for math learning. *Learning Technology newsletter*, 6(3):13–15, July 2004. accepted.

[9] OASIS Inc. Oasis creates tc to define service oriented architecture (soa) reference model, Feb 2005. See `http://xml.coverpages.org/ni2005-02-08-a.html`.

[10] M. O'Neill. Xml and web services: Are we secure yet?, July 2004. See `http://www.ftponline.com/javapro/2004_07/magazine/features/moneill/`.

[11] K. Porayska-Pomsta and H. Pain. Providing cognitive and affective scaffolding through teaching strategies. In *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS)*. Springer, Berlin, 2004.

[12] P. Prescod, Feb 2002. See `http://webservices.xml.com/pub/a/ws/2002/02/20/rest.html` and also `http://www.prescod.net/rest/`.

[13] R. Schmidt. Jsr00056: Java network launch protocol, May 2001. See `http://www.jcp.org/aboutJava/communityprocess/first/jsr056/`.

[14] The Globus Alliance. The ws-resource framework, Feb 2005. See `http://www.globus.org/wsrf/`.

[15] C. Ullrich, D. Wallach, and E. Melis. What is poor man's eye tracking good for. In E. O'Neill, P. Palanque, and P. Johnson, editors, *17th Annual Human- Computer Interaction Conference 2003, Volume 2*, Swindon, 2003. British Computer Society.

[16] J. Zimmer and M. Kohlhase. System description: The mathweb software bus for distributed mathematical reasoning. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction*, number 2392 in Lecture Notes in Artificial Intelligence. Springer Verlag, 2002.