ActiveMath: System Description*

Erica Melis, Eric Andres, Georgi Goguadse,
Paul Libbrecht, Martin Pollet, Carsten Ullrich
DFKI Saarbrücken, D-66123 Saarbrücken, Germany

ActiveMath is a web-based, user adaptive learning environment that employs content encoded in a semantic XML-representation of mathematical knowledge and integrates several mathematical service systems. In a nutshell we describe the distributed web-architecture, the knowledge representation, the integration of external systems, and the dynamic generation of the course presentation. A demo is available at `http://www.mathweb.org/activemath/demo`.

The ActiveMath system dynamically generates interactive (mathematical) documents according to the user's content needs and presentation preferences and provides a generic distributed architecture and facilities for a user interaction with (mathematical) service systems. It supports instructional as well as exploratory learning and leaves freedom of navigation and responsibility to the user.

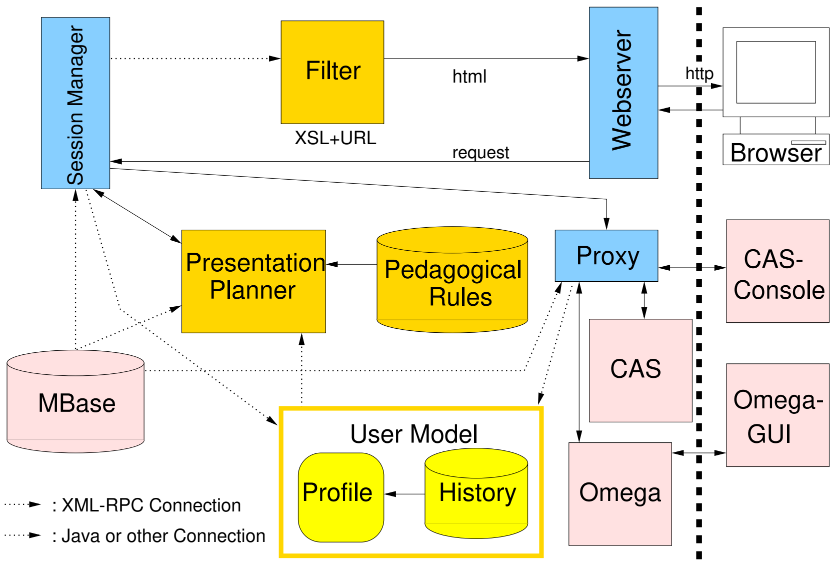

Figure 1: Architecture of ActiveMath

Figure 1 depicts the architecture of ActiveMath, i.e. its components and the communications between them. In a nutshell, the figure shows a client-server web-architecture with a browser at the client side. Currently, ActiveMath integrates the following components: a session manager, the knowledge base MBase [3], a presentation planner (see §2), a user model, a pedagogical module, and deduction and

*The project which led to the results presented in this paper was funded by the German Bundesministerium für Bildung und Forschung, contract 01AK008/2. The authors are responsible for the content of this publication.

computation service systems such as the proof planner of ΩMEGA [6] and the Computer Algebra System (CAS) MAPLE [2]. The components can communicate over the Internet by a standardized XML-RPC protocol[7]. In Figure 1, the arrows represent the data flow between the components. For instance, requests of the user and (in the other direction) HTML-pages are communicated via a web-server. The presentation planner requests and processes information from MBASE, from the user model, and from the pedagogical module in order to generate a document adapted to the user's goals, preferences, and knowledge. Information about the user's actions, such as the history of her actions and the time intervals of her reading a concept or the success of solved problems, is passed from the session manager to the user model (as well as from the proof planner and CAS to the user model), where it is used for updating.

1 Knowledge Representation

Our knowledge representation `OMDoc` [5], is an extension of the `OpenMath` [1] standard. `OMDoc` encodes the semantic of mathematical objects as well as meta-data and mathematical facts such as theorems, definitions, proof methods, and proofs and includes natural language formulations as well as formal (`OpenMath`) objects. `OMDoc` uses a semantic XML-based representation of mathematical knowledge that provides an *ontology* for the content of the course which is indispensable for a reuse of teaching and learning material and for a combination of material from different sources. In addition to the actual conceptual content, our knowledge representation contains meta-data for structures, dependencies, and pedagogical information which can be used for the dynamic generation of interactive documents.

2 Presentation Planning

The presentation planning dynamically constructs a course that suits the learner's goals (e.g. to learn everything needed to understand the mathematical concept *groups*), preferences (e.g. colorful presentation, prefered language), and mastery level.

The presentation planner employs the expert system JESS [4] to dynamically assemble the course for each user according to the given pedagogical rules and user model. Examples for simple pedagogical rules (in a readable format) are the following:

IF omega=no	THEN	remove(omega-exc)
IF scenario(examPrep) AND uk(?C,low)	THEN	addExc(?C)=(1,1,2,3)
IF scenario(preBook) AND pres(detail)	THEN	pattern=(motiv,concept,exm,exc)

The first rule expresses that if the user is not able to use ΩMEGA, then no interactive exercises using it will be presented. The second rule determines that if the user wants to prepare for an exam and her knowledge of the current concept that instantiates the variable ?C is low, then four exercises of difficulty 1, 1, 2, and 3 will be presented. The third rule determines the pattern motivation, concept, examples and exercises for the presentation of concepts if the user wants to work on a detailed document in the scenario prerecorded book.

The document generation process has several stages: (1) Mathematical content is retrieved from MBASE. All the MBASE items on which a requested concept depends are collected. (2) The collection of content items is then processed according to the information in the user model and the pedagogical module. For example, exercises at the suitable level of difficulty are chosen. In this step, interaction facilities are chosen too, e.g., the availability of a CAS for solving a task. This step returns an instructional graph with IDs of MBase-items. (3) This step determines additional interaction facilities, such as buttons for the use of ΩMEGA or MAPLE, according to the user model and pedagogical knowledge. (4) Then the instructional graph is linearized. (5) Eventually, the session manager fetches the actual OMDoc content for each item in the linearized instructional graph. Filters (e.g. XSL style sheets) are applied and transform the OMDoc content to HTML.

3 External Systems

An external system acts as an expert service in the learning environment. The benefit of integrating it is manyfold. (1) Expert services allow the user to explore a problem interactively. (2) It allows the user to focus on a particular skill to solve a specific problem. For example, the user of the system should not fail to calculate the limit of a function just because he cannot find the factorization of a polynomial; a computer algebra system may serve this task. Similarly, the user should not fail to find the overall proof just because he cannot prove a simple subgoal. It may suffice from a pedagogical point of view that the user suggests some intermediate goals that should be provable and a proof planner fills the gaps in the proof plan. (3) They can check the correctness of the user's calculation or derivation. (4) The system can help with feedback of different sorts, e.g. on where it is promising to explore and where dead ends are reached.

The XML-RPC communication allows to issue commands and their parameters (such as the problem to be solved and the methods that should be available) to the service via a proxy. In the other direction information about the (interactive) problem solving can be communicated from the service. This information can be used to update the user model, to give feedback, or to inform a teacher.

References

[1] O. Caprotti and A. M. Cohen. Draft of the open math standard. Open Math Consortium, `http://www.nag.co.uk/projects/OpenMath/omstd/`, 1998.

[2] B.W. Char, G.J. Fee, K.O. Geddes, G.H. Gonnet, and M.B. Monagan. A Tutorial Introduction to MAPLE. *Journal of Symbolic Computation*, 2(2):179–200, 1986.

[3] A. Franke and M. Kohlhase. Mbase: Representing mathematical knowledge in a relational data base. In F. Pfenning, editor, *Proc. 17th International Conference on Automated Deduction*, Lecture Notes on Artificial Intelligence. Springer-Verlag, 2000.

[4] E. Friedman-Hill. *Jess, the Java Expert System Shell*, 2000. `http:/herzberg.ca.sandia.gov/jess/`.

[5] M. Kohlhase. OMDoc: Towards an internet standard for the administration, distribution and teaching of mathematical knowledge. In *Proceedings AISC'2000*, 2000.

[6] E. Melis and J.H. Siekmann. Knowledge-based proof planning. *Artificial Intelligence*, 115(1):65–105, November 1999.

[7] UserLand Software. XML-RPC specification, `http://www.xmlrpc.org/spec`.