Semantic-Aware Components and Services of ActiveMath

The ActiveMath group:
Erica Melis, Giorgi Goguadze, Martin Homik,
Paul Libbrecht, Carsten Ullrich, Stefan Winterstein
DFKI Saarbrücken, Germany
Email: dev@activemath.org

Abstract

ActiveMath is a complex Web-based adaptive learning environment with a number of components and interactive learning tools. The basis for handling semantics of learning content is provided by its semantic (mathematics) content markup which is additionally annotated with educational metadata. Several components, tools, and external services can make use of that content markup, for instance a course generator, a semantic search engine, and user input evaluation services. The components and services have to communicate, pass content and state changes, actions, etc including mathematical semantics and educational markup. The novel event infrastructure supports this communication. This article focuses on usage of the content's semantics by selected novel components and sketches the communication.

Keywords: semantic e-Learning, semantic services, semantically annotated learning content

Introduction

The broad accessibility and interaction of Web resources and Web-services requires the reuse and interoperability of content resources and of services. Our research in several ActiveMath related projects contributes to these general goals of the Semantic Web in various ways. Some of the results are described in this paper.
ActiveMath is a complex and rather mature Web-based adaptive learning environment for mathematics. Its open architecture integrates a number of components, tools and services. The components are modularly implemented and different types and sources of knowledge, such as content, exercises, knowledge about the learner, knowledge about the context, pedagogical strategies, tutorial strategies, etc reside in different places (databases). This eases the modification and reuse of knowledge as well as configurability considerably.
The components and Web-services provide abstract interfaces and can therefore be employed by different applications, ie reused. The same functionality can be used by various systems. For instance, different Web-services and components can rely on the same learner model service. Similarly, the content stored in several (distributed) repositories can be used.
Currently, the functionalities of ActiveMath's components (see Figure 1) and services include:

- a tutorial component comprising of an adaptive course generator and a reactive suggestion component (tutorial component)
- semantic retrieval of learning objects (mediator)
- updating beliefs about the learner (learner model)
- playing and evaluating interactive exercises (exercise subsystem, Goguadze, González-Palomo and Melis, 2005)
- search for content and semantics as well as dependencies (semantic search)
- efficient rendering and delivery of content (presentation subsystem)
- evaluation of mathematical expressions in user's input (computer algebra systems)
- interactive concept map tool

- learner management
- input of mathematical expressions (mathematical input editor).

[Insert Figure 1 here]
Since this paper cannot cover all of ActiveMath's novel and interesting features and components, it focuses on a few and confines mostly to Semantic Web aspects, which include the semantic knowledge representation, its usage by some components and services, and a new part of the communication between the server.
The main contributions show which services are necessary for Semantic Web learning environments, which underlying semantic knowledge representation we consider appropriate and why, how some components and services make use of that knowledge representation and how the previously existing XML-RPC communication of components and services is augmented with a novel event notification framework.
The article is organized as follows. First, the semantic knowledge representation that underlies ActiveMath, OMDoc, is very briefly reviewed (for more details see Melis et al., 2003). Then, selected services and components are presented mostly from the Semantic Web point of view. This includes the knowledge base, the mediator, the tutorial component, the presentation component, and one of the client-based interaction tools. Each component/tool is described briefly, with a focus on those features that are particularly related to semantic encoding and the usage of semantic information. Finally, a new part of the communication between components and services is described.

Semantic Content Encoding

One objective of using a generic semantic markup language is to keep the encoded content reusable and interoperable by other, even non-educational, mathematical applications. ActiveMath uses the semantic XML-markup language for mathematical documents, OMDoc (Kohlhase, 2001), for its content encoding. OMDoc has evolved as an extension of the OpenMath (European) standard for mathematical symbols and expressions (Caprotti and Cohen, 1998). ActiveMath uses these representations for the following reasons: they provide a semantics for mathematical expressions and a standardized structure of the content. Especially OpenMath is used for a variety of mathematical services and thus, all applications can take advantage of each other’s advances in tools and implementations.
ActiveMath’s content is represented by a collection of typed items called “learning objects” annotated with metadata. The semantic information includes types, relations, and other mathematical and educational metadata. The type indicates a characterization of the items as collection, theory, concept or satellite items: an OpenMath symbol defines a mathematical concept abstractly; a theory assembles concepts and it can import other theories; concepts (definitions, algorithms, and assertions/theorems) are the main items of mathematical contents, whereas satellites (exercises, examples, explanations, introductions) are additional items of the content which are related to one or several concepts. All items are accessible via a unique identifier. The content (including the OpenMath symbols) induces an ontology.
For ActiveMath, we have extended the primarily mathematical OMDoc to serve educational purposes. For this, the metadata characterize not only mathematical, organizational, and intellectual property rights properties such as Dublin Core, but also educational annotations. These LOM compatible metadata include properties such as difficulty, learning context, and field.
Moreover, items can be linked by relations. The *domain_prerequisite* relation expresses a mathematical dependency between concepts, ie, it describes which concepts are mathematically necessary in order to define the current one. The *for* relation links satellite items to concepts and definitions to symbols. For instance, an example or exercise may illustrate or train a related definition. The *against* relation describes an item as a counter

example or misconception of the related concept. The *is_a* relation serves to represent mathematical hierarchies and special cases, eg, a “function” is a specific “relation”.
Note that several (mathematically equivalent) definitions may exist *for* one symbol, eg, the convergence of a function can be defined via ε-δ fomulae or via sequences. Definitions and theorems can be different for different learning-contexts or fields too. That is, only the Open Math/ symbol layer of the induced ontology is abstract in the sense that it represents concepts uniquely as it required in certain ITS applications. ActiveMath does not yet handle different *versions* of one and the same element which have the same metadata (eg, language, learning context).
As we shall see in the following, the types and metadata properties and relations of the learning objects can be exploited to perform very advanced dynamic adaptation of the learning material, sequencing and presentation to the learner's needs and the context. Why is the semantic representation of symbols, ie, Open Math, then needed in addition?

(I) In ActiveMath the evaluation of the learner's input for questions and exercises can “subcontract” Computer Algebra Services, CAS, (possibly different CAS, whichever is suitable). For this, any of the CAS service has to “understand” the meaning of mathematical expressions, which is realized via the standardized semantic representation OpenMath. The language of a specific CAS or presentation MathML is not sufficient for that purpose.

(II) Based on the unique OpenMath Symbol reference, ActiveMath' presentation component can generate symbol presentations that are appropriate for the needs and context of the learner, eg, country-dependent notation and various output formats (HTML+MathML, PDF or SVG). This is a feature unique to ActiveMath currently.

(III) In ActiveMath, copy&paste of formulars is possible based on references to symbols' semantics hidden in the presentation. This is a feature which is unique for ActiveMath.

Selected Components and Services of ActiveMath

The ActiveMath Web-architecture is open in the sense that new components and services can be accommodated and communicated with. Interfacing between components is based on local interfaces, remote Web-service interfaces, and on a generic event messaging mechanism described later. An advantage of this architecture is the generic integration of external as well as internal tools and the asynchronous exchange of information between components and services. The configuration mechanism can refer to core components as well as to services that are globally and permanently available.

Knowledge Base Services

ActiveMath clearly separates the content from the functionalities working on it (such as course generation and presentation). MBase is a database that serves as a generic storage for the OMDoc-documents in ActiveMath. Queries include the retrieval of XML-content/items based on identifiers and search based on relations between items in both directions. Currently, fuzzy and semantic search are supported by this database.
At load time, MBase resolves references relative to the mathematical theory and collection the element is in, into absolute references that include the theory and collection explicitly. This makes the structuring more flexible, integration more robust, and eases modifications. For communication, the MBase service provides an XML-RPC protocol. This was chosen to balance expressivity and performance.
The integration of other, third-party knowledge bases is supported by the mediator service

described in following section.

Mediator Service

Content in a Web environment may be distributed over several repositories and even be described with different metadata schemas. Therefore, it is important that components can abstract from the knowledge sources. They should not need to locate the relevant sources; neither interact with each source separately nor manually combine the results.

ActiveMath achieves this separation of concerns by using the mediator approach to information integration (for a recent overview on this topic, see Doan, Noy and Halevy, 2004). That is, a special service acts as a mediator between the knowledge processing and knowledge storing components. The mediator offers a uniform query interface to a multitude of autonomous data sources.

Semantic search of the mediator is realized by searching for types and metadata. A typical query to the ActiveMath mediator inquires about the existence of elements that fulfill given constraints. An answer consists of a list of identifiers of matching elements. For instance, the query *(getItems (class exercise) (property for derivation) (property difficulty low) (property field computer-science))* returns all easy exercises for the mathematical concept *derivation* with a computer science context.

[Insert Figure 2 here]

Figure 2 illustrates the query processing. First, a client component passes a query formulated in its own schema to the mediator (1). The mediator translates the query into the mediated schema (2) and then subsequently for each knowledge source from the mediated schema into their respective schemas (3). The queries are send to each knowledge base (4), and the answers (lists of identifiers, 5) merged together (6). Finally, the mediator sends the answers back to the client (7).

The mediated schema is based on an ontology of instructional objects. Because existing metadata standards such as IEEE LTSC LOM (2002) fail to represent sufficient information about the resources to enable complete automatic processing, we developed an ontology that describes different types learning objects from an instructional point of view (Ullrich, 2005).

[Insert Figure 3 here]

Figure 3 provides an overview of the instructional ontology. It is important to note that the ontology is orthogonal to OMDoc and does not describe the domain taught by the learning material, eg, “addition of fractions”. Instead, each class of the ontology stands for a particular instructional role a learning object can play, for instance “An *Example* for addition of fractions” or “An *Introduction* for addition of fractions”.

Several design goals influenced the structure of the ontology:

- Domain independence. The classes of the ontology should be independent of the domain that is being taught, i.e., they should characterize learning material about mathematics as well as literature.
- Pedagogical flexibility. The classes should be independent of the instructional theory underlying the learning material, i.e., they should describe traditional didactic as well as more constructivist approaches (eg, based on PISA’s literacies, OECD, 1999, as realized in the LeActiveMath project).
- Completeness. The classes should cover the range of learning materials as much as possible.
- Compatibility. Existing standards and e-learning knowledge representations should be mapped on the classes as easy as possible.
- Applicability. Authors should be able to understand and apply the ontology. LOM, for instance, is notorious for putting a heavy load on content developers.
- Efficiency in searching. Users should be able to narrow the search down quickly using terms that reflect what they need.
- Machine processability. The classes (together with other metadata) should enable sophisticated machine applications to find learning objects without human intervention.

Central to the ontology is the distinction between *Fundamentals* and *Auxiliaries*. Fundamentals subsume instructional objects that describe the central pieces of knowledge. Auxiliary elements include instructional objects which provide additional information or activities about the fundamentals.

In the scope of the European Network of Excellence “Kaleidoscope” (http://www.noe-kaleidoscope.org), two case studies illustrated the benefits of an ontological approach in the area of data mining (Merceron, Oliveira, Scholl and Ullrich, 2004). Additionally, we experiment with integratin several German (math) repositories and will report the result elsewhere.

The mediator can be extended to handle on the fly construction of learning objects by generators: The mediator treats a generator similar to any other data source. A query is translated as described above and send to the generators. The generators check whether they can generate a learning object respecting the constraints, and, if so, send back identifiers from which the learning object can be reconstructed later when it is presented to the learner.

Generalization of Course Generation

Course generation automatically assembles learning objects to a greater unit that supports the learner to reach a given learning goal. Today’s course generation uses rather simplified pedagogical criteria, eg, to select those learning objects with the lowest typical learning time of a learning object (Karampiperis and Sampson, 2005). However, to generate a course adapted to the individual learner’s goals and needs requires more elaborate mechanisms.

This section first describes ActiveMath approach to course generation, which is based on the declarative representation of generic pedagogical knowledge, and then continues to show how this approach meets today’s challenges posed by the Semantic Web.

Representing and Executing Pedagogical Knowledge

Assembling pedagogically adequate courses requires sophisticated knowledge. Such knowledge can be represented and executed in a number of ways, for instance as planning operators (van Marcke, 1992, Vassileva, 1995) or, as in the previous ActiveMath course generation, as rules for an expert system (Ullrich, 2003).

The current version of ActiveMath uses the hierarchical task network (HTN) planner JShop2 (Ilghami, 2005). HTN-planning is a very efficient planning technique and offers a relatively straight-forward way for representing human expert knowledge (Erol, Hendler and Nau, 1994).

HTN incorporates heuristic knowledge in the form of decomposition rules: A planning problem is represented by sets of tasks; *methods* decompose non-primitive tasks into sub-tasks until a level of primitive tasks is reached, which can be solved by *operators*. In the following examples, the notation is as follows: variables have the prefix “?”, and operators the prefix “!”.

A pedagogical task corresponds to the goals one wants to achieve and combines the two dimensions domain and educational goal. Formally, a task t is defined as a tuple $t=(l,c)$, where l is a tutorial objective and c a list of unique identifiers of learning objects (content goals).While c specifies the concepts the course will primarily target, the tutorial objective determines the kinds of learning objects selected for c.

A top-level task that serves as a starting point of course generation is *teachConcept id*. The goal of this task is to assemble a structured sequence of learning objects that help the learner to understand the content goal c. Using different collections of tasks and methods (ie, different tutorial strategies), this task can be planned differently. Hence, the task-based approach can serve to represent a variety of pedagogical strategies.

[Insert Figure 4 here]

Figure 4 illustrates parts of the representation of a strategy based on Merrill’s “First principles of instruction” (Merrill, 2002). The upper method is applicable on the task *teachConcept x*. It

links to the learner's existing knowledge by teaching all unknown prerequisite concepts (task *teachPrerequisitesConcepts ?c*). For each prerequisite concept *y*, a method (not shown) inserts the task *teachSingleConcept y*. Teaching a single concept should happen the following way: the first learning objects serve to motivate the student and to familiarize him with the concept. Then, the concept itself is presented, followed by additional elaborations. Several examples and exercises serve to demonstrate and train application of the concept. Each section closes with concluding remarks.
[Insert Figure 5 here]
While the above methods result in sequences of several learning objects, other methods may select only a single element. A frequently occurring task of this kind is to select satellite learning objects taking information about the user into account. Figure 5 shows a subset of the methods that select an exercise.
The methods encode best possible and alternative ways of selecting satellite elements. An *ideal* method takes a larger number of user properties into account and poses a larger number of constraints on the satellite element. For instance, the upper method of Figure 5 checks among others whether the user is highly motivated. Then, it searches for an exercise of a competence level one level higher than the student can master. Additionally, the learning context of exercise is constraint to the educational level of the learner, and, similar, its field should correspond to the field of the learner.
In case no ideal method is applicable, *fallback* methods come into play. They encode less constraining conditions on the learning objects and serve to insert elements in case no ideal method can be fulfilled. The bottom method of Figure 5 matches if any exercise for concept *?c* exists that is of the correct competency level and has a learning context equal or lower than the educational level of the learner.
Several extensions were necessary in order to adapt HTN-planning to a Web service environment. Most importantly, content is distributed over the Internet and made available via Web servers and learning object repositories. Integrating this content by course generation involves the difficulty that traditional AI-planning requires to evaluate a method's preconditions against the planner's world state. However, in a distributed environment this would require to mirror all the metadata of the content stored in the repositories. This is simply infeasible. Therefore, we extended the HTN-planner so that queries about learning objects in a method's preconditions result in a call to the mediator.
JShop2's *call* command allows integrating external function call during planning. The function call *GetElements conditions* queries the mediator to search the repositories for elements that fulfill the given conditions. *assignIterator ?p ?identifiers* generates all bindings of *?p* to the elements of the returned identifier list. Thereby subsequently all possible values can be tried when backtracking.
The result of the planning is a sequence of learning objects called *course structure*. Similar to the "organization" element of an IMS Content Package (IMS, 2002), it consists of nested sections with the leaves being pointers to learning objects. It is important to note that because tasks represent a vast range of pedagogical issues, the size of the generated courses ranges from a single element to a complete curriculum.

Integrating Support Services in Course Generation

A wide variety of systems that support the learning process in various ways have been developed. For instance, some stimulate meta-cognition (White and Shimoda, 1999) and others perform assessment (Conejo et al., 2004). A course should integrate these services, not arbitrarily but in a pedagogically sensible way: during the learning process, at specific times the usage of a tool will be more beneficial than at some other time. For instance, reflecting upon the learned content may be most effective at the end of a lesson.
In case these services are encapsulated in a learning object, the previously described methods are expressive enough to add them to a course. Otherwise, the integration happens by *service-adding* methods. These methods encode the pedagogical knowledge at what situation in the

course the learner should use a tool and insert calls to the tool at the appropriate place in the course structure. When the learner navigates through the course, an invitation to use the service is presented to her.
Additionally, the planner needs to take into account whether a service is available or not, depending on the configuration of the learning environment and on its server status. This required extending the planner so that the method's preconditions can check for the availability of a service. In case the service is available, its call is inserted in the course. Otherwise, an alternative method is applied. This way, the pedagogical knowledge remains reusable, regardless of the actual configuration.
[Insert Figure 6 here]
Figure 6 illustrates the integration of the assessment tool Siette (Conejo et al., 2004). The upper, ideal method checks whether Siette is available and if so, inserts a call to the service into the course structure. In the fallback case, the assessment will be simulated. A short text is added in the course that describes that the following exercises will assess his knowledge, and five exercises are inserted. The updated method *detailedguidedTour* (lower part of the Figure) includes the call to Siette as a sub-task.

Course Generation as a Service

The course generator makes its functionality available as a service to other components of ActiveMath by using a set of *public* tutorial objectives. Any component can send an event with a task comprising of a public tutorial objective and learning object identifiers to the course generator and will receive a course structure as a result.
An often used public task is *insertAppropriateExercise identifier*. Several services make use of this task, eg, the NextBest Suggestor (Melis & Andres, 2005) that monitors the learner's interactions with the system and, in case a problem is diagnosed, offers remediating feedback. The agent focuses on the diagnosis and sends a request to the course generator to determine the exact content to present. Thus, services only decide which the tutorial task is adequate for the current situation and then delegate its execution.
In order to avoid that the learner is flooded with suggestions in case several components decide to present them at the same time, an additional component, the tutorial control, collects incoming tutorial task requests, selects which one to process next and sends it to the tutorial planner. The content structure calculated for the task is then presented to the learner and, if he wishes, included into the course. The course generator, together with the tutorial control and suggestors such as NextBest, form the tutorial component of ActiveMath.

Concept Map Service

The interactive concept map tool CMap is a client service application that is available to ActiveMath. Students can use it to exercise the construction of a structure of a mathematical domain and to discover relations between (mathematical) concepts. CMap's user interface provides an (interactive) visualization in which nodes represent concepts (and sometimes other learning objects) and edges between nodes represent relationships between concepts (or other learning objects). Nodes and edges can be introduced with the help of a dynamically generated menu. A node's name can be copied from ActiveMath presentations which carry hidden identifiers and, thus, semantics. Alternatively, a node's name can be provided through handwriting.
In the concept map tool, the evaluation of the learner's input substantially relies upon the ontological relations of the content (in OMDoc) and on the semantic encoding of mathematical symbols/concepts. In order to generate feedback, the edges introduced as a connection between two nodes by a student are checked against the relations available for the content in the knowledge base MBase and in authored exercises. The (hidden) semantics of the nodes makes the user's input machine-readable and checkable. Currently, the handwritten names of nodes cannot be evaluated since they do not carry any semantics.

Communication between CMap and ActiveMath is twofolded: synchronous requests to MBase and asynchronous client-events. CMap publishes client-events via ActiveMath's event framework (see next Section). The learner model can listen to those events and process the event information to update its belief about the learner. CMap publishes exercise events only. It publishes appropriate action step events together with the step's action information to the event handler. Appropriate steps are those steps that are necessary for the reconstruction of the performance value. Finally, an *exercise finished* event notifies the learner model and conveys information about the learner's performance.

Presentation service

ActiveMath features a modular presentation subsystem (Ullrich, Libbrecht, Winterstein and Mühlenbrock, 2004). The reasons for its development have been

- the potential re-usability of the component
- ease of extension and modification of such a component
- efficiency and performance.

The presentation component receives a collection of OMDoc elements and applies several transformations in a row. For instance, the transformation selects the language the OMDoc element should be presented in, etc. OMDoc representations of mathematical symbols are transformed to their MathML, HTML+Unicode and LaTeX equivalents according to their semantics determined by OpenMath. Similarly, transformations produce different results for different target output formats such as HTML+MathML, HTML+Unicode, SVG, Latex/PDF. For these, the transformed elements are cached format-dependently for performance reasons. After that intermediate caching, additional transformations are applied adaptively. For instance, adaptive link annotation can happen at this stage.

The transformations insert semantic information that is invisible after the final rendering: mathematical expressions and all subterms are attached with an identifier that points to all the semantic information and, therefore, is the basis for copy & pasting semantic information and interoperability of services on the presentation.

Communication of Components and Services

Previously, ActiveMath's Web-services were based on the XML-RPC communication protocol because of its familiar semantics, its simplicity, and very broad platform support. For many situations and applications however, a remote procedure call (ie, synchronous communication) is inappropriate. In particular, some components need just a quick notification when events of interest occur in another service or component. Moreover, the sending component may not be aware of all services which need to be informed.

The components interested in such messages may not be known in advance or can change over time and constellation. To simplify a flexible integration and reuse of Web-services, a mechanism for registering interest and for propagating events is needed. Therefore, the Web-service approach is complemented by an event framework in ActiveMath now.

Events are a mechanism for a powerful and flexible, yet rather loose integration of components. The event paradigm is especially useful for state update notifications and reactions to them. Similar messaging and notification frameworks exist for SOAP. Since we use XML-RPC rather than SOAP for simplicity and performance reasons, we devised a new lightweight event framework.

An example for event "publication" is the following: when the learner finished working on an exercise, the exercise subsystem issues an event. The event carries information describing the learner, the identifier of the exercise, the success rate, the time stamp of the event, etc. Listeners to such an event can be the learner model as well as the suggestor of the tutorial component.

A component that *publishes* events is an *event source*. A component that *subscribes* to the

events published by an event source is called a *listener* which receives *event messages* from the event source.
In contrast to a full-fledged messaging model, events are not sent from a specific sender to a specific recipient but remain anonymous: when publishing an event, the component is usually not aware who is listening to the events (only the module managing the subscriptions is). Also, usually the listener does not care which component or module created the event, it only knows where to subscribe to the events it is interested in.
Each event message object consists of attributes. The attributes common to all ActiveMath events are:

- a type, indicating what happened (eg, "user X has logged in", "new content available"). Depending on the type, the event may carry additional data for further describing the event.
- a timestamp, indicating when it happened: Events happen at a single point in time (and space). The timestamp indicates the wall-clock time of the event source when the event took place.
- a source, indicating where happened. This is typically the name of the producer, ie, the class name of the component that produced the event.

Other attributes are specific to the type of events.
It is often desirable to group event types across several dimensions, eg, for filtering ("I want all event concerning a user", "Give me all events related to the Dictionary"). However, events don't fit well into a rigid type hierarchy: For instance, consider the even "user logged in". What would be the "natural" group of such an event? An application event? An user event? An interaction event? What's needed is a flexible, non-hierarchical type framework that allows for a mixing of characteristics such as "is associated to a user". Therefore, ActiveMath uses event tags. Tags are labels for event types. Each event type can be associated to zero or more tags. Tag definitions can be nested, that is, a tag can inherit from another one. Multiple inheritance is allowed.
Tags can add attributes to an event. For example, the "user" event tag adds the attribute *userId* to an event. Tags defined without attributes (such as "application") just serve as marker tags. The following attributes are found in event tags, among others: user event tag (identifier of the user that caused this event, empty if user is anonymous); session event tag; item event tag (the identifier of the item, interaction and application event tag.

Remote Events

Remote eventing based on XML-RPC has been implemented for both server-to-server and rich-client-to-server communication. A special case is the event exchange with the browser client. Our solution is based on asynchronous client-server communication using the browser's *XMLHttpRequest* object. For XM-RPC, the attribute values are simple XML-RPC datatypes. Therefore, attribute names or an event type must be unique across the event type's class, the event tags, and the event base class.
Each event publishing Web-service needs a wrapper including a Remote Event Publisher. For the communication of (mathematical) semantics the wrapper needs to include a conversion of the service's internal representation to OpenMath (often called phrasebook).

Related Work

A number of research groups investigate the usage of Semantic Web techniques in an educational context. However, to our knowledge, no other system covers such a broad and complete set of functionalities as ActiveMath does.
For instance, only few systems (eg, Cuypers and Sterk, 2001) offer a semantic representation of the content, which is the basis for rendering different output formats and for using the content as input for tools such as Computer Algebra Systems.

Today, many systems use XML-based metadata, based on and extending IMS or IEEE LOM (IEEE, 2002). Several groups, unsatisfied by the limited pedagogical coverage of LOM worked on extensions of LOM or special sets of metadata expressive enough for pedagogically mature systems (Rodriguez-Artacho andVerdejo, 2000, Pawlowski, 2002, Buendía-García and Díaz-Pérez, 2003, Cisco, 2003, Lucke, Tavangarian and Voigt, 2003, Swertz, 2003). The ontology of instructional objects described in this article draws on these approaches and represents the minimal pedagogical information necessary for pedagogically sound learner support.
Adaptive Hypermedia (AH) covers techniques of adapting hypermedia objects with respect to the user. Well known systems are ELM-ART (Weber and Brusilovsky, 2001) and AHA (de Bra, Aerts, Smits and Stash, 2002). ActiveMath offers several AH techniques but is more flexible as complete courses can be generated based on pedagogical knowledge. Course Generation (or Adaptive Trail Generation) has been investigated since long (Vassileva, 1995, van Marcke, 1998, Specht and Oppermann, 1998, Steinacker, Seeberg, Reichenberger, Fischer, and Steinmetz, 1999) (at that time without Semantic Web techniques) and is still an active area of research. For instance, Keenoy et al. (2004) and Karampiperis and Sampson (2004), focus on the generation of learning trails and courses. However, they do not allow for such detailed representation of pedagogical knowledge as done in ActiveMath.
In the last years, the Semantic Web challenges for education were investigated by several groups. Neidl et al. (2002) describe an RDF-based peer to peer networking infrastructure. Miklos, Neumann, Zdun and Sintek (2003) describe an approach to query Semantic Web resources by querying and transforming RDF models using a rule language based on Horn logic. They focus on general approaches, thus the e-learning scenario provided as an example is limited to querying exercises. Conlan, Lewis, Higel, O'Sullivan and Wade (2003) and Dolog, Henze, Nejdl, and Sintek, (2004) describe approaches to adaptivity, which are more restricted than the one by ActiveMath.

Conclusion

This article describes selected ActiveMath research and development that is strongly related to semantic representation of educational content, to semantics-based Web-services and Web-communication. In particular we describe how the course generator, the mediator service, the CMap tool service, and the presentation component deal with the semantic information and how services and components communicate events.
As opposed to some purely academic research in this area, we care about performance issues and not only about proof-of-concept. We think that this makes a difference in the Semantic Web world.
Currently, one of the bottlenecks for a large amount of shared content is the authoring of semantically and metadata-annotated content and of interactions. Although content is available for several mathematical domains and in several languages and authored by a number of partners, authoring still holds several open research problems, among them: reduction of the workload for authors by semi-automatic annotation with metadata; support of collaborative Web-authoring; and more flexibility of rendering semantically encoded content.
The semantic encoding of educational content by several parties can lead to additional problems in the case where different parties rely on different (mathematical) domain ontologies for the same domains or sub-domains. This can either be avoided by restricting authors to a particular domain ontology or – more flexibly and technically demanding – by mapping the contents to an abstract domain ontology and additionally providing reasonable versioning. Currently, we are investigating these solutions. The situation here is very different from what happens for content encoding in intelligent tutoring systems which provide much smaller shares of a subject domain, mostly focus on exercises, and one or few pedagogists whose learning objects rely on a unique domain ontology are care carefully designed. Opening up authoring to many authors gives rise to new questions.

Acknowledgement

This publication partially is a result of work in the context of the LeActiveMath funded under the 6th Framework Program of the European – (Contract IST-2003-507826). The authors are solely responsible for its content.

References

De Bra, P. & Aerts, A. & Smits, D. & Stash, N. (2002). AHA! Version 2.0 – more adaptation flexibility for authors. In *Proceedings of the AACE ELearn'2002 conference*.

Buendía-García, F. & Díaz-Pérez, P. (2003). A framework for the management of digital educational contents conjugating instructional and technical issues. *Educational Technology and Society*. 60(4), 48–59.

Caprotti, O. & Cohen, A. M. (1998). Draft of the open math standard. Open Math Consortium.

Cisco Systems, Inc. (2003). *Reusable learning object strategy: Designing and developing learning objects for multiple learning approaches*.

Conejo, R. & Guzman, E. & Millan, E. & Trella, E. & Perez de-la Cruz, J. L. & Rios, A. (2004). Siette: A web-based tool for adaptive teaching. *International Journal of Artificial Intelligence in Education*. 14, 29–61.

Conlan, O. & Lewis, D. & Higel, S. & O'Sullivan, D. & Wade,V. (2003). Applying adaptive hypermedia techniques to semantic web service composition. In *Proceedings of AH2003: Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems*. 53–62.

Cuypers, H. & Sterk, H. (2001). Mathbook, web-technology for mathematical documents. In *Proceedings of the BITE 2001 conference*.

Doan, A. & Noy, N. F. & Halevy, A. Y. (2004). Introduction to the special issue on semantic integration. *SIGMOD Rec*. 33(4), 11–13.

Dolog, P. & Henze, N. & Nejdl, W. & Sintek, M. (2004). Personalization in distributed elearning environments. In *Proc. Of the Thirteen International World Wide Web Conference*.

Erol, K. & Hendler, J. & Nau, D. S. (1994). HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Vol. 2. 1123–1128. AAAI Press/MIT Press.

Goguadze, G. & González-Palomo, A. & Melis, E. (2005). Interactivity of Exercises in ActiveMath, *Proceedings of International Conference on Computers in Education (ICCE 2005*. In Print.

IEEE Learning Technology Standards Committee (2002). *1484.12.1-2002 IEEE standard for Learning Object Metadata*.

Ilghami, O. (2005). *Documentation for JSHOP2*. Technical Report CS-TR-4694, Department of Computer Science, University of Maryland.

IMS Global Learning Consortium (2004). IMS content packaging information model, version 1.1.3.

Karampiperis, P. & Sampson, D. (2005). Adaptive instructional planning using ontologies. In *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies, ICALT 2004*. 126–130.

Keenoy, K. & Poulovassilis, A. & Christophides, V. & Rigaux, P. & Papamarkos, G. & Magkanaraki, A. & Stratakis, M. & Spyratos, N. & Wood, P. (2004). Personalisation services for self e-learning networks. In Koch, N. & Fraternali, P. & Wirsing, M. (Eds) *Proc. of 4th International Conference on Web Engineering*, volume 3140 of *Lecture Notes in Computer Science*. 215–219. Springer.

Kohlhase, M. (2001). OMDoc: Towards an internet standard for mathematical knowledge. In Lozano, E. R. (Ed) *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2000*, LNAI. Springer Verlag.

Lucke, U. & Tavangarian, D. & Voigt, D. (2003). Multidimensional educational multimedia

with <ML>3. In *Proceedings of E-Learn 2003, World Conference on E-Learning in Corporate, Government, Healtcare, & Higher Education.* 101–104.
Melis, E. & Andres, E. (2005). Global feedback in ActiveMath. *Journal of Computers in Mathematics and Science Teaching.* 24, 197–220.
Melis, E. & Büdenbender, J. & Andrès, E. & Frischauf, A. & Goguadze, G. & Libbrecht, P. & Pollet, M. & Ullrich, C. (2003). Knowledge Representation and Management in ActiveMath. *Annals of Mathematics and Artificial Intelligence.* 38(1-3), 47-64.
Merceron, A. & Oliveira, C. & Scholl, M. & Ullrich, C. (2004). Mining for content re-use and exchange – solutions and problems. In *Poster Proceedings of the 3rd International Semantic Web Conference, ISWC2004.* 39–40.
Merrill, M.D. (2002). First principles of instruction. *Educational Technology Research & Development.* 50(3), 43–59.
Miklos, Z. & Neumann, G. & Zdun, U. & Sintek, M. (2003). Querying semantic web resources using TRIPLE views. In *Proceedings of the 2nd International Semantic Web Conference.*
Nejdl, W. & Wolf, B. & Qu, C. & Decker, S. & Sintek, M. & Naeve, A. & Nilsson, M. & Palmér, M. & Risch, T. (2002). Edutella: a P2P networking infrastructure based on RDF. In *WWW '02: Proceedings of the 11th International Conference on the World Wide Web.* 604–615, New York, NY, USA: ACM Press.
Organisation for Economic Co-operation and Development (1999). (Ed) *Measuring Student Knowledge and Skills – A New Framework for Assessment.*
Pawlowski, J. M. (2002). Modellierung didaktischer Konzepte mit dem Essener-Lern-Modell. In *Proceedings of Workshop Neue Medien in der Bildung: Standardisierung im e-Learning.*
Rodriguez-Artacho, M. & Verdejo, M.F. (2000). High-Level design of web-based environments for distance education. *Computers and Education in the 21st century.* 275–286. Kluwer Academic Publishers.
Specht, M. & Oppermann, R. (1998). ACE - adaptive courseware environment. *The New Review of Hypermedia and Multimedia.* 4, 141–162.
Steinacker, A. & Seeberg, C. & Reichenberger, K. & Fischer, S. & Steinmetz, R. (1999). Dynamically generated tables of contents as guided tours in adaptive hypermedia systems. In *Proceedings of the EdMedia & EdTelecom.* 167–175.
Swertz, C. (2003). Das didaktische Metadatensystem DML. In Schwaenzel, R. (Ed) *Sharing Knowledge: Scientific Communication. Proceedings of 9. Kongresses der Informations- und Kommunikationsinitiative der wissenschaftlichen Fachgesellschaften Deutschlands.*
Ullrich, C (2003). *Pedagogical rules in ActiveMath and their pedagogical foundations.* Seki Report SR-03-03, Universität des Saarlandes, FB Informatik.
Ullrich, C. (2005). The learning-resource-type is dead, long live the learning- resource-type! *Learning Objects and Learning Designs.* 1(1), 7–15.
Ullrich, C. & Libbrecht, P. & Winterstein, S. & Mühlenbrock, M. (2004). A Flexible and Efficient Presentation-Architecture for Adaptive Hypermedia: Description and Technical Evaluation. *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies*. 21-25.
Van Marcke, K. (1992). Instructional expertise. In Frasson, C. & Gauthier, G. & McCalla, G.I. (Eds) *Proceedings of the Second International Conference on Intelligent Tutoring Systems*, number 608 in Lecture Notes in Computer Science . 234–243. Springer.
Van Marcke, K. (1998). GTE: An epistemological approach to instructional modeling. *Instructional Science.* 26, 147–191.
Vassileva, J. (1995). Dynamic courseware generation: at the cross point of CAL, ITS and authoring. In *Proceedings of ICCE'95* . 290–297.
Weber, G. & Brusilovsky, P. (2001). ELM-ART: An adaptive versatile system for web-based instruction. *International Journal of Artificial Intelligence in Education.* 120 (4), 351–384.
White, B.Y. & Shimoda, T.A. (1999). Enabling students to construct theories of collaborative inquiry and reflective learning: Computer support for metacognitive development.

International Journal of Artificial Intelligence in Education. 10, 151–182.

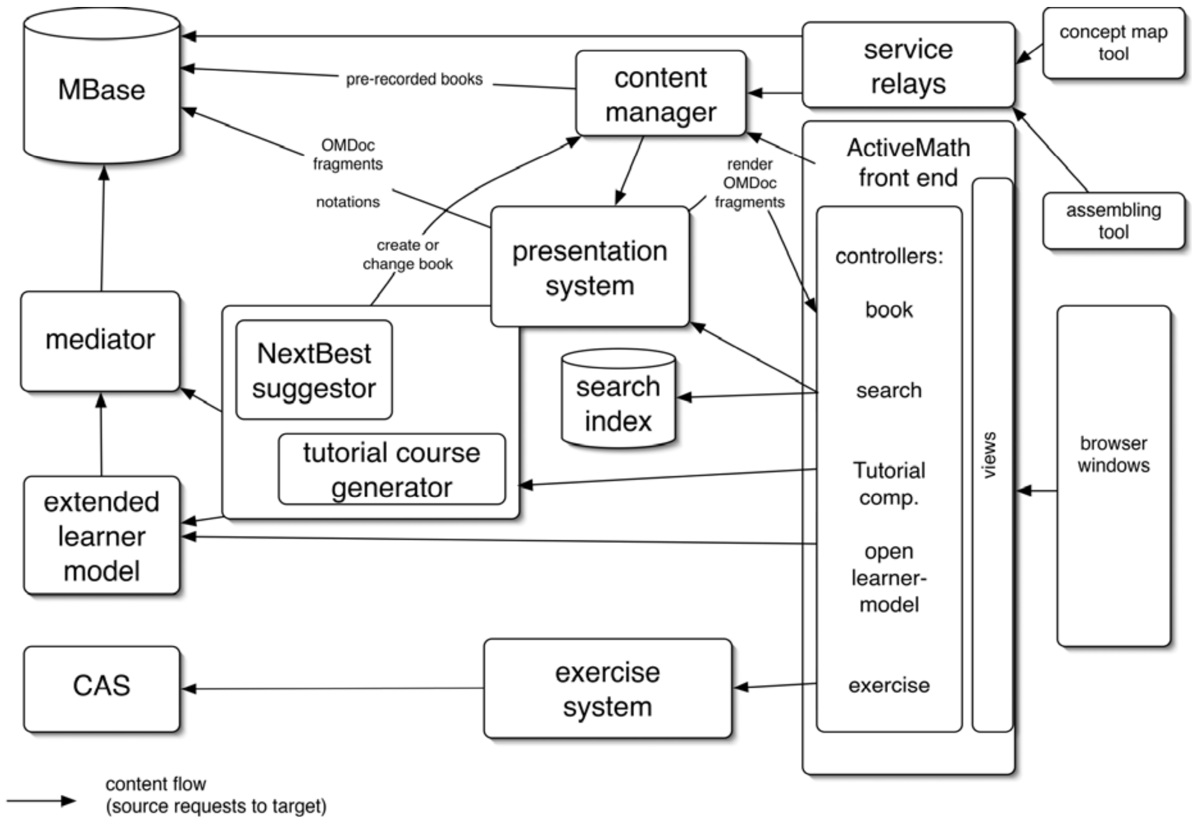

Figure 1. A<span>CTIVE</span>M<span>ATH</span> *architecture*

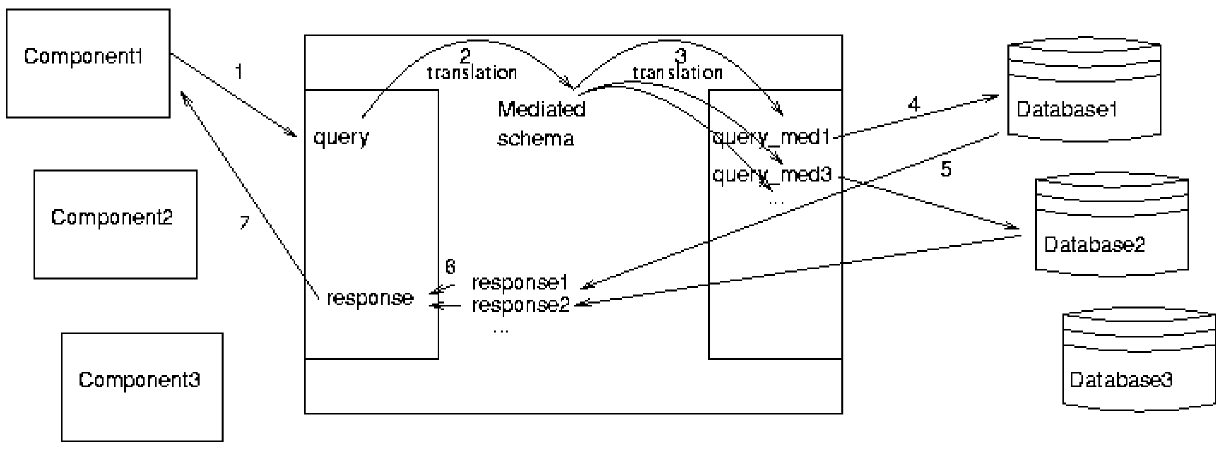

Figure 2. General mediator architecture

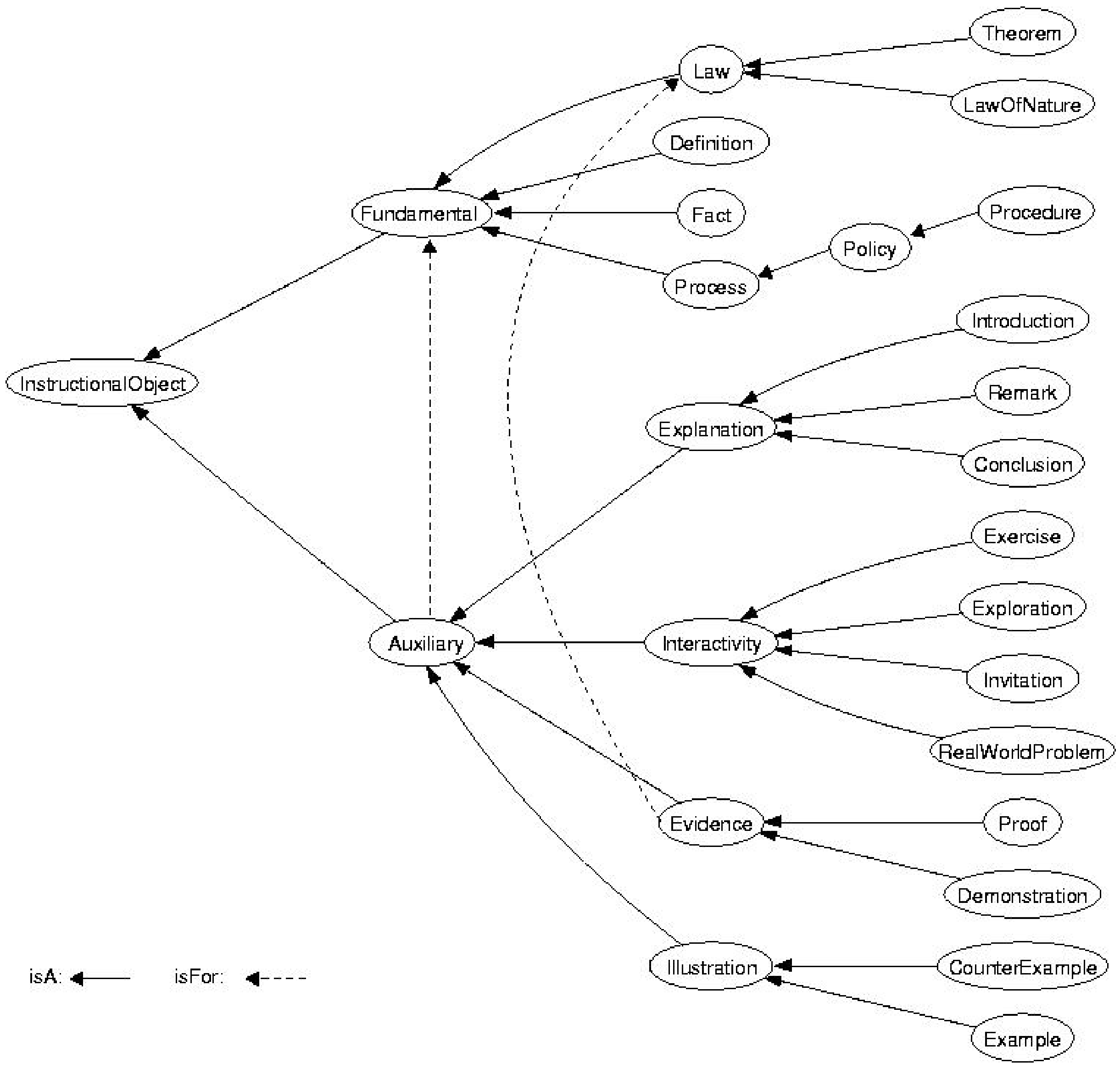

Figure 3. The ontology of instructional objects

```
(:method (teachConcept ?c)
        () ;; no precondition
        (;; sub-tasks:
                (!startSection NewBook ?c)
                (teachPrerequisitesConcepts ?c)
                (teachSingleConcept ?c)
                (!endSection)))

(:method (teachSingleConcept ?c)
        ()
        (;; sub-tasks:
                (!startSection ?c)
                (tryMotivate ?c)
                (tryIntroduce ?c)
                (insertElement ?c)
                (tryExplain ?c)
                (selectAppropriateExamples ?c)
                (selectAppropriateExercises ?c)
                (tryConclude ?c)
                (!endSection)))
```

Figure 4. Teaching a concept by a detailed guided tour

```
(:method (selectAppropriateExercise ?c)
       ;; ideal method: if motivation is high, then select
       ;; exercises with higher competency level
       (;; preconditions:
        (learnerProperty field ?field)
        (learnerProperty educationalLevel ?el)
        (learnerProperty motivation ?m) (>= ?m 3)
        (learnerProperty competencyLevel ?c ?cl)
        (assignIterator ?exercise
                      (call GetElements ((class exercise)
                                         (relation for ?c)
                                         (property learningcontext ?el)
                                         (property competencylevel (+ 1 ?cl))
                                         (property field ?field)))))
       (;; sub-task: (insertElement ?exercise)))

(:method (selectAppropriateExercise ?c)
         ;; alternative method: select exercise from any field
       (
        (learnerProperty allowedEducationalLevel ?el)
        (learnerProperty competencyLevel ?c ?cl)
        (assignIterator ?exercise
                      (call GetElements ((class exercise)
                                         (relation for ?c)
                                         (property learningcontext ?el)
                                         (property competencylevel ?cl)))))
       ((insertElement ?exercise)))
```

Figure 5. Selecting an exercise

```
(:method (assess)
       ;; if Siette is available, then use it for assessment
       ((call ServiceAvailable Siette))
       ((!insertServiceCall Siette)))

(:method (assess)
       ;; fallback: insert some exercises
       ()
       ((insertText assessment)
        (insertExercises 5)))

;; updated method for teachingProblemByGuidedTour
(:method (teachConcept ?c)
       ()
       ((!startSection NewBook ?c)
        (assess) ;; insert a call to the assessment service
        (teachPrerequisitesConcepts ?c)
        (teachSingleConcept ?c)
        (!endSection)))
```

Figure 6. Integration of an assessment service