ActiveMath – a Learning Platform With Semantic Web Features

Erica Melis, Giorgi Goguadze, Paul Libbrecht, Carsten Ullrich
DFKI and Universität des Saarlandes, Saarbrücken, Germany

June 7, 2009

Abstract

ActiveMath is an intelligent e-Learning platform that exhibits a number of Semantic Web features. Its content knowledge representation is a semantic XML dialect for mathematics, semantic search is enabled, some of its components work as a web service and, vice versa, it employs certain foreign web services, e.g., for diagnostic purposes.

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

•**EdNote!** *20 seiten max*

1 Introduction

ActiveMath was one of the first educational systems that seriously addressed the Semantic Web – such as semantic representation and metadata – in a realistic e-Learning application. It is around for quite some time now and has evolved from a prototype to a full-blown platform that is used by an international community centred in Germany so far.

ActiveMath has typical intelligent tutoring system's (ITS) components such as domain (expert) model, a student model, and pedagogical modules comprising course generator, tutorial strategies, and feedback generators. They are organized as a client-server application with additional web-services.

What is rather untypical for traditional ITSs are the advanced features that make it a Semantic Web application, e.g., truly semantic markup and reuse of content, semantic search, interoperable content and components, distributed architecture, generation of web presentations from the representation of content, asynchronous event framework, etc. Hence, ActiveMath is also a workbench for studying benefits of combining ITS and semantic e-Learning technologies as suggested in [Brooks et al.(2006)]. Different from traditional ITSs, ActiveMath encodes the domain model, i.e., the domain ontology, *implicitly* in the content stored in a knowledge base. Because of this encoding and an active community of authors, the content and thus the ontology/domain model can evolve and change over time. Hence, ActiveMath has to take care of those changes preferably in an automatic way rather than in a constant re-engineering effort.

In the following, we describe ActiveMath's relevant design parts including knowledge representation, architectural issues, communication, role of ontologies, as well as relevant features of some ActiveMath components among them web-services. In order to provide a rather self-contained chapter, we summarize some of ActiveMath's features which were included more detailed in [Melis et al.(2006)].

2 Design Principles and Preliminaries

ActiveMath has been designed with web-communication, interoperability, semantic knowledge representation and metadata standards in mind.

2.1 Architecture and Communication

From the beginning, ActiveMath had a client-server architecture [**?**] as depicted in Figure 1. Its components communicate via http requests •**EdNote!** or use an even more elaborate communication via a mediator or broker as described below. An event system (not depicted in Figure 1) enables an asynchroneous communication of components (including external systems) to which components and external systems 'issue' event-information and components can 'listen'. For instance, the exercise subsystem or an external player can send information about the student's performance in an exercise and the student model may listen and use this information.

Paul, George, true for all communication??; kind of.. there's always inside-vm components... they are not http-bound (e.g., as below, the search to the content store

The figure shows the server of the ActiveMath platform in the middle and its web-service communications with external servers at the left-hand-side.

Note that the employed ontologies are not incuded in Figure 1 because they are not handled as main components but their information is implicit in the representation of content (stored in content repositories) and in external content dictionaries of `OpenMath`, which are introduced below. However, the course generator uses an explicit stable ontology of instructional objects, oio, as explained below. Several content knowledge bases can be accessed by the course generator by way of mapping their ontologies of instructional objects to the course generator's oio [Kärger et al.(2006)]. A domain ontology mapping is not (yet) used. The external diagnostic systems communicate with ActiveMath via a broker and exchange information in `OpenMath` as detailed below. The search component communicates with the content base(s) via direct java calls and initiates an indexing upfront.

ActiveMath's course(ware) generator is called Paigos [Ullrich(2008)], and it uses information about learning objects, the learner and his/her learning goals to generate an adapted sequence of learning objects that supports the learner in achieving his goals. Hence it communicates with the student model and

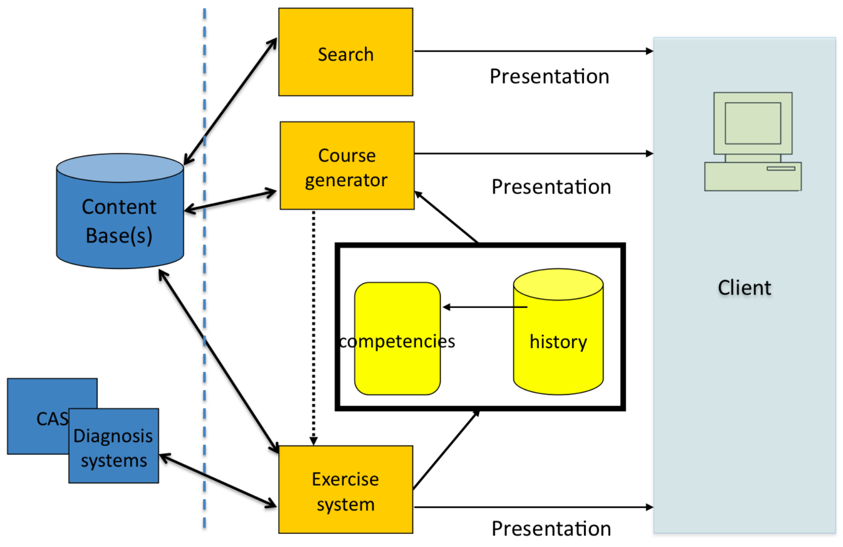

Figure 1: Coarse architecture of ActiveMath with services

the content base(s). PAIGOS is based on an extensive model of expert teaching knowledge – about 300 "rules" define how to assemble different types of courses. PAIGOS can function as a service and be accessed by other learning environments. Section 2.2.1 describes what is required of the knowledge representation to enable such a service, and Section 3.3 describes the service in detail.

The student model provides information to the other components via an interface. The presentation subsystem (multiply presented in Figure 1 to allow for a clearer depiction) takes IDs, fetches XML lerning objects (LOs) from the content base(s) and generates the actual presentations which the client browser will present. The actual generation uses presentation information from the user profile/request, the course generator, exercise system, and the system settings. This information communication is not depicted in the architecture figure to keep it clear.

2.2 Semantic Knowledge Representation

The knowledge representation in ACTIVEMATH is based on the `OMDoc` standard for mathematical documents [Kohlhase(2006)]. It defines fine-grained learning objects (LOs) connected to each other by relations and annotated with metadata. In ACTIVEMATH, we differentiate between two types of `OMDoc` LOs: (1) so-called concepts (also called 'knowledge components' in ITS publications) that are the main elements of the ontology such as symbols representing abstract mathematical concepts, definitions of these concepts, axioms, theorems and proofs; (2) satellite elements such as example, exercises and types of texts that elaborate on, explain, or train the related concepts.

The OMDoc format itself uses OpenMath [OpenMath] as embedded format for representing mathematical formulaæ. OpenMath is a well-established standard for representing mathematical formulaæ and, indeed, a semantic representation. Its main goal is the interoperability of formulaæ and, thus, of the systems which process them. It defines the mathematics symbols' semantics by the usage of so-called *content dictionaries* [OpenMath], which contain agreed upon symbol declarations providing a hook, to which symbol occurrences point to when using an OpenMath symbol (OMS) element. This enables a semantic evaluation and interoperability of mathematical formulæ. The symbol declarations are complemented by a description in regular English and by formal properties which are mathematical statements that should hold for the symbol to be interoperable. Mathematical documents contain not only formulaæ but also several types of text, links, and potentially multimedia parts. Therefore, OMDoc extends OpenMath: all textual fragments can occur in multiple languages and be interleaved with formulæ; structural elements such as definitions, examples, exercises, etc. are added – as usual in XML – including text etc.; and reference/grouping constructs, such as omgroup and theory.

•**EdNote!** The grouping construct theory models formal mathematical theories and allows for a rich management of namespace in the usage of imports.

Georgi, Theorie nur erlautern, falls theory spaeter vorkommt, sonst nicht, weil verwirrend!

OpenMath content dictionaries serve as reference for symbols rather than for document elements. Referencing the content dictionay or grouping, to which a symbols belongs adds semantical information to mathematical expressions, e.g., whether + is an operation for real numbers or for matrices. This provides additional information for the expressions' semantic evaluation by external mathematical reasoning services and for the diagnosis of user input.

•**EdNote!**

Georgi, das folgende habe ich gestrichen, weil es ablenkt. es koennte bestenfalls als footnote durchgehen

2.2.1 Ontologies

As sketched above, most ontological information is represented in the content, rather than explicitly in separate ontologies. This design decision goes back to ActiveMath's flavor of an open platform to which (evolving) content can be added/modified by an authors' community. This implies frequent changes of the ontologies except the rather stable oio. For a particular instance of ActiveMath we are currently testing an approach in which the domain ontology is a separate OWL-formalization to which the content and its metadata will refer to.

ActiveMath uses two different ontologies, a domain ontology and a pedagogical one. The domain ontology contains abstract concepts (symbols) and their definitions, theorems, axioms as concepts and describes the subject domain from a mathematical point of view, e.g., relations between concepts that indicate the equivalence of two definitions.

The pedagogical ontological information is represented in our extension of OMDoc. It includes relations such as prerequisite-for •**EdNote!** and proper-

Georgi, ?

ties of LOs such as difficulty as, e.g., defined in the LOM standard. Moreover, the pedagogical information declares the type of a learning objects according to its instructional function and the OIO ontology. This is independent of the specific (mathematical) subject domain.

We had to develop the additional ontology OIO because existing learning object metadata standards such as LOM [IEEE LTSC (2002)] failed to describe learning objects sufficiently precise for intelligent components to integrate them automatically into the students' learning work flow. For instance, LOM's `learning-ResourceType` mixes pedagogical and technical or presentation information: while its values `Graph`, `Slide` and `Table` describe the format of a resource, other values such as `Exercise`, `Simulation` and `Experiment` cover the instructional type. They represent different dimensions, hence need to be separated for an improved decision making. Furthermore, several instructional objects are not covered by LOM (e. g., `definition`, `example`). As a result, LOM fails to represent the instructional types sufficiently precise to allow for *automatic* usage of learning objects, in particular, if it involves complex pedagogical knowledge necessary for effective learning support. For instance, LOM has no easy way to determine to what extent a resource annotated with `Graph` can be used as an example.

The then novel *ontology of instructional objects* (OIO) [Ullrich(2005)] contains this previously missing information. Its classes are shown in Figure 2. The root class of the ontology is `instructionalObject`. Central to the ontology is the distinction between the classes `fundamental` and `auxiliary`. The class `fundamental` subsumes instructional objects that describe the central pieces of domain knowledge (concepts). Auxiliary elements include instructional objects which contain additional information about the fundamentals as well as training and learning experience. The OIO enables several of ActiveMath's advanced pedagogical features amd it allows to define the course generation knowledge such that it is independent of the specific domain.

Applications of the ontology in areas other than course generation were investigated in the European Network of Excellence Kaleidoscope and published in [Merceron et al.(2004)]. Moreover, the OIO was used for a revised version of the ALOCoM ontology [Knight et al.(2006)], in the e-Learning platform e-aula [Sancho et al.(2005)], and in the CampusContent project of the Distant University Hagen [Krämer(2005)].

The OIO ontology facilitates the process of making third-party repositories available to the course generator which can assemble a course from resources of different repositories. The OIO helps to provide the course generator's functionality as a service to systems that register their repositories (described in Section 3.3). In [Rostanin et al.(2006)] we showed that it can also be applied to completely different domains (e.g., for work flow embedded e-Learning).

2.2.2 Metadata

Metadata used by ActiveMath can be divided in three main categories: general administrative metadata, mathematical metadata, and pedagogical meta-

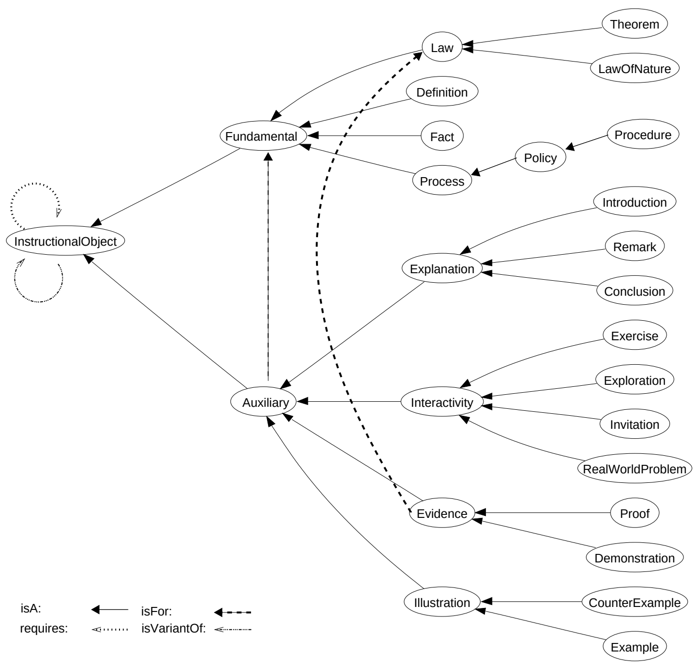

Figure 2: Overview of the Ontology of Instructional Objects

data.

For general annotations of LOs such as `title` of the item, `date` of its creation, name(s) of `author(s)`, copyright information and so on ActiveMath uses the standard Dublin Core metadata element set. The `Rights` element/values used for specifying copyright is replaced by the standard Creative Commons metadata.

Mathematical metadata define mathematical types of LOs such as definition, theorem, exercise and relations between them. There are several kinds of mathematical relations between LOs. The most frequently used are: (1) the `domain_prerequisite` relation that indicates that a concept is needed in order to introduce the current concept and (2) the `for` relation indicating that a LO relates to a concept to define, explain, illustrate, introduce, prove, or train a concept.[1] •**EdNote!**

alle: ist das FOR wirklich mathematisch?? oder niocht doch pedagogical? Paul: it's both

Pedagogical metadata include some metadata imported from LOM, such as `learning_context, difficulty, field`, and `abstractness`. They define parameters of 'auxiliary' LOs that help the components of ActiveMath to act intelligently and to model the student appropriately.

`Competency` metadata are assigned to 'auxiliary' LOs. Following the approach of Anderson and Krathwohl [Anderson et al.(2001)], a competency is represented as a pair of a cognitive process and one or more domain concepts.

[1] Typically, a cognitive task analysis should be the basis for introducing the `for` for exercises.

This metadata defines a skill the LO addresses. ActiveMath can relate to several competency schemes, such as Bloom's Taxonomy of Learning Goal Levels, the PISA competencies [OECD(2004)], and an extension of Anderson and Krathwohl's scheme described in [Melis et al.(2008)].

•**EdNote!**

Georgi: relation to standard technology such as SCORM, PAPI, IMS-LD!! see PL's input commented

3 Web-Services and Components of ActiveMath

All components and communications in Figure 1 use the decribed knowledge representation, most importantly, the exercise subsystem with its diagnosis, the course generator, and the student model.

3.1 Diagnostic Services

Diagnosis is a basis for generating feedback in interactive exercises, which is an efficient alternative to authoring (limited) feedback.

ActiveMath has a generic framework for distributed diagnostic services. This framework implements interfaces for connecting different kinds of remote services using existing protocols supported by web applications. ActiveMath can query two kinds of services for diagnosing the student's input: (generic) computer algebra systems (CASs) and domain reasoner services which rely on formalizations of human-like reasoning and possibly frequently used incorrect rules in a specific mathematical domains.

The semantic `OpenMath` markup for mathematical formulaæ and a generic format for queries to the diagnostic services support the interoperability of different CASs and reasoners (see below). ActiveMath can communicate with CAS and domain reasoner services which have so-called phrasebooks translating `OpenMath` formulaæ into the actual syntax of the service and vice versa translating computation result back to `OpenMath` for ActiveMath. Due to the fact, that the `OpenMath` format represents semantics of mathematical formulaæ, such phrasebooks can always be implemented nd already exist for a number of CAS, e.g., for Maple [Maple], Mathematica [Mathematica], Maxima [Maxima], Yacas [2], WIRIS [1].

Currently, ActiveMath integrates and communicates with the following CASs: YACAS, Maxima, and WIRIS. Figure 3 shows different ways of connecting to CASs that we realized in our framework, which are dependent on a CAS's implementation: the WIRIS server is connected to ActiveMath via XML-RPC and has an internal `OpenMath` phrasebook. YACAS has native support for `OpenMath` and is communicating directly via an internal `OpenMath` protocol. The Maxima server communicates via WDSL and the queries are piped through an external phrasebook. Currently, the most frequently used CAS connected to ActiveMath is YACAS, since it is modular. easily extensible, and open source. New domains can be attached to YACAS by extending domains that are represented as modules in form of scripts that can be attached as parameters to the YACAS process or loaded into the running system on fly.

CASs are very efficient and fast in providing diagnoses needed for the generation of a flag feedback (correct/incorrect feedback) as well as for a correct solution of a given problem.

CAS services are also used for creating so-called randomized exercises, in which the complete solution of an interactive exercise is parametrized. For every admissible instantiation of the parameters a concrete exercise and its solution can be generated. The Randomizer of the exercise subsystem of ActiveMath generates exercises by instantiating the parameters with randomly chosen values from defined ranges over sets (of numbers or functions) and intervals. Since the solution of each step of a problem is represented as a mathematical expression, for each randomized exercise the student's answers can be diagnosed as correct or incorrect by a CAS. For more information about the Randomizer component see [Dudev, González Palomo(2007)].

More diagnoses can be obtained when a domain reasoner is available for the mathematical domain(s) of the exercise, e.g., errors in a soution, the student's solution strategy, irrelevant steps. A domain reasoner can respond to queries which are used to generate hints for the learner such as

- next step
- correct input for current step
- number of steps to final solution, etc.

An example of a domain reasoner service is SLOPERT [Zinn(2006)], which encapsulates expert and buggy human-like rules for the (mathematical) domain of symbolic differentiation. This service maintains an internal state and, thus, can trace the (partial9 solution of the student and diagnose his/her errors. Another domain reasoner connected to ActiveMath is MathCoach [Grabowski et al.(2005)], which is, however, stateless and cannot trace a student's solution. A series of domain reasoners is being developed at Open Universiteit Netherlands [Heeren at al.(2008)] using Haskell They can respond to queries similar to those of ActiveMath (see section 3.1.2.

Currently, ongoing work is implementing rule-based domain reasoners in the form of YACAS modules, that could provide more sophisticated stepwise diagnosis and are answering ActiveMath specific queries described in the section 3.1.2.

3.1.1 Service Query Architecture

ActiveMath implements a novel service architecture for the diagnosis of student's actions in mathematical problem solving. A broker architecture distributes queries to external diagnosis *services* as shown in Figure 3. The `Query Broker` accesses those services that are registered for the (mathematical) domain needed for the diagnosis. This domain is recognized by analyzing the included expressions and `OpenMath` symbols. •**EdNote!** For instance, a domain reasoner for symbolic differentiation is only queried for (sub-)problems in symbolic

Georgi, stimmt das?

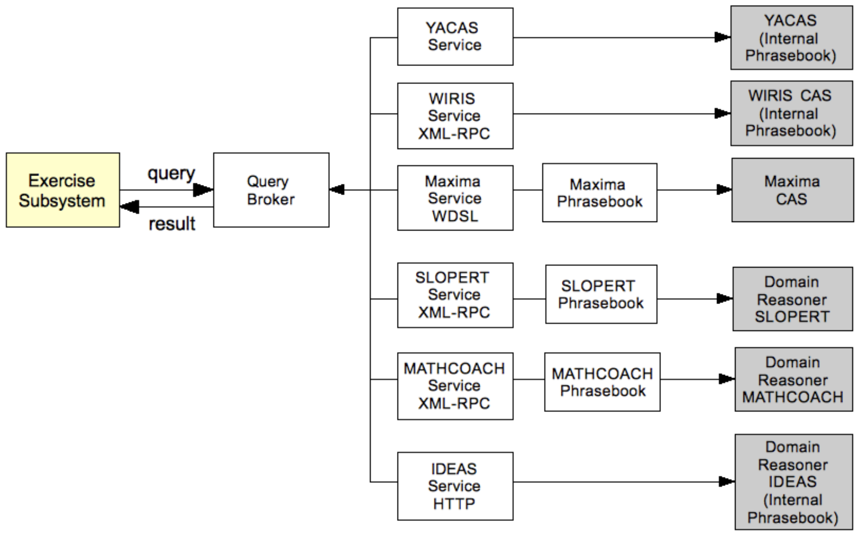

Figure 3: Diagnosis framework architecture

differentiation. The subscribed mathematical services themselves can also send a query back to the `Query Broker` in case a subexpression belongs to another domain and has to be analyzed by another reasoner. For example, a domain reasoner for symbolic differentiation can send a query back to the broker if it needs to simplify an arithmetic expression. The `Query Broker` passes this new query to a CAS or an arithmetic domain reasoner.

Few other systems try to make mathematical services provers accessible through the web. Examples of such are MONET services [MONET(2003)], or MathServe [Zimmer, Autexier(2006)].

3.1.2 Queries

In ActiveMath generic queries are used to access any diagnosis service. The queries include a number of dimensions, one of them is *context*, a novel construct.

A context defines (sub-)sets of rules and functions that a domain reasoner or a CAS is allowed to use for a response to the query. The background for this restriction is that the student's learning situation determines which 'rules' and functions he/she is supposed to employ. That is, usually a student input such as `Solve`*(expression)* (where `Solve` is a CAS command) wont be accepted even though it is semantically equivalent with a correct result. For instance, if the task of the student is to differentiate the function $f(x) = (x + 1) \cdot x$. If the student has not yet learned the product rule, a reasonable and correct next step would be an arithmetic transformation that removes brackets. Using the product rule would not be expected from the student. In this case, the evaluation of the student's answer needs to exclude the product rule from the context but include the arithmetic context.

In order to formalize queries used for diagnosis we defined a format for queries including :

- **action** of the query with the commands explained below
- (list of) **input expressions** to be evaluated or compared with each other and depending on the action.
- **context** of action identifying the set of applicable rules, e.g., arithmetic, differentiation, logic
- **number** of iterations defines how many atomic steps the domain reasoner should perform in the given context.

In the following, e, e_1, e_2, are `OpenMath` expressions, C is a context of a query, N is the number of iterations. A solution path is a list of results of consecutive rule applications, which are annotated with rule identifiers.

Currently the following queries to diagnostic services are used in ActiveMath:

- query(`getResults`, e, C, N) - returns the list of final nodes of all paths of length N starting at e in the context C
- query(`compare`, e_1,e_2, C, N) - returns true if there exists a path of the length N from e_1 to e_2 in the context C, false otherwise
- query(`getRules`, e, C) - returns the list of the identifiers of expert rules applicable to e in context C
- query(`getBuggyRules`, e_1, e_2, C, N) - returns the list of the identifiers of all buggy rules that belong to a path from e_1 to e_2 in the context C. This query is possible for those domain reasoners that can reason with (typical) buggy rules and some CASs, which can be extended to do so.
- query(`getUserSolutionPaths`, e_1, e_2, C, N) - returns the list of all paths of length N from e_1 to e_2 in the context C
- query(`getExpertSolutionPaths`, e, C, N) - returns the list of all paths of length N starting at e in the context C. In this query C can consist of expert rules only.
- query(`getNumberOfStepsLeft`, e, C) - returns the number of steps left to reach the final node of the shortest expert solution path in context C
- query(`getRelevance`, e_1, e_2, C) - returns 'true' if the expression e_2 is closer than e_1 to the actual solution in the context C,

For example, a query for information about *the next two steps for computing derivative of the function $f(x) = (x+1) \cdot x$ using only arithmetic simplifications and differentiation rules except for a product rule* looks as follows:

$$\text{query}(\texttt{getResults},\ (x+1)\cdot x,\ C,\ 2),$$

where C is the composite context formed by concatenating arithmetical context and differential rules without product rule.

3.2 Student Model

In most educational content, the metadata related to competencies refer to one of (the standard) taxonomies. For instance, the PISA specification for mathematics 'competencies' includes `think, argue, model, solve, represent, language, tools`. Hence, to achieve interoperability a student model needs to be able to adopt different frameworks for competencies/skills which are used in educational contents/systems. ActiveMath's *semantics-aware student model* (SLM) is flexible enough to act upon contents using different competency frameworks. Currently, it can choose between the competency taxonomies used in PISA [OECD(2004)], Bloom's taxonomy [Bloom(1956)], and the mmore recent two-dimensional taxonomies as described in [Anderson et al.(2001), Melis et al.(2008)].

Moreover, a system that relies on evolving content produced by a community of authors ActiveMath needs to adapt to (frequent) modifications of the domain model as well. Therefore, the structure of the SLM is dynamically generated from the metadata represented in the content representation.

Becasue of the focus of the paper, we will mainly describe the build-process of SLM rather than go into detail of the updating of competency-values through Item Response Theory and Transferable Belief Model [Faulhaber and Melis(2008)]. Note, however, that metadata such as the difficulty and competency-level of the recent exercise influences the updating process: difficulty provides a parameter of IRT.

The generation of the student model is data-driven. The structure of SLM consists of nodes, each for a single concept to estimate competencies for. SLM automatically creates a node for each concept k included in the current learning content of a student, e.g., the concept 'definition of fraction' or the rule 'addition of fractions with unlike denominators'. See Figure 4. SLM stores each associated competency value $m(k,p)$ within the node, where a competency is defined as a pair (k,p), in which p is a cognitive process, such as *apply an algorithm* or *model* a mathematical problem that is applied to k. For one-dimensional competency frameworks such as PISA or Bloom's SLM translates the competencies to the two-dimensional framework. Inter-node relations are dynamically extracted from the content metadata, most importantly the `prerequisite` relationship, which determines propagation in SLM. See Figure 4 for an illustration.

The student model communicates its results to ActiveMath's event system that every other component can listen.

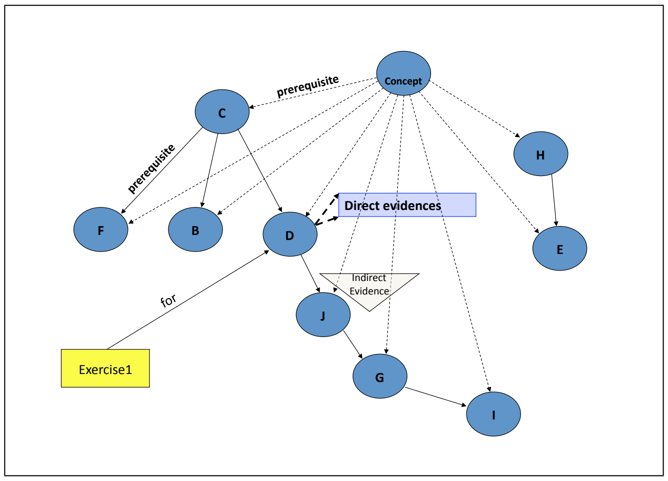

Figure 4: Structure of the student model and its propagation links

3.3 Course Generation Service

The course generator of ActiveMath (Paigos) is designed as an component whose services can be accessed by other learning environments, too. This requires that Paigos and those learning environments share a common understanding of the type of courses that are to be generated as described in Section 3.3.1. Then, Section 3.3.2 describes the communication between Paigos and its clients.

3.3.1 Representations

Pedagogically sensible course generation requires reasoning about learning scenarios. In traditional course generation systems, scenarios consist only of learning objects, which represent the target content that is to be learned. Such an approach ignores that also selection and sequencing of LOs depend on the course's purpose. For instance, LOs in a course for preparing an exam may differ from LOs in a guided tour.

Furthermore, in the Web of today, where systems are no longer standalone but embedded in the eco-system of the Web, the representation of the scenarios should enable communication about and exchange of scenarios between different systems and services. Thus, the representation needs to contain sufficient semantic information to enable such functionality.

Van Marcke [Van Marcke(1998)] introduced the concept of an *instructional tasks*, which represents an activity that can be accomplished during the learning process. This helps to define scenario more accurately since both, the content and the instructional task are essential aspects of a learning goal. Therefore, we define scenarios as a combination of the two dimensions *content* and *task*.

A *scenario* is a tuple $t = (p, L)$, where p is an identifier of the instructional

Identifier	**Description**
`discover`	Discover and understand concepts in depth
`rehearse`	Address weak points
`trainSet`	Increase mastery of a set of concepts by training
`guidedTour`	Detailed information, including prerequisites
`trainWithSingleExercice`	Increase mastery using a single exercise
`illustrate`	Improve understanding by a sequence of examples
`illustrateWithSingleExample`	Improve understanding using a single example

Table 1: A selection of tasks used in PAIGOS

task and L is a list of learning object identifiers. L specifies the course's target concepts, and p influences the structure of the course and the learning objects selected.

For instance, the instructional task to discover and understand content in depth is called `discover`. Let's assume that `def_slope` and `def_diff` are the identifiers of the learning objects that contain the definition of the mathematical concept "average slope of a function" and "definition of the differential quotient", respectively. The scenario for a learner who wants to discover and understand these two concepts is $t =$ (`discover`, (`def_slope`, `def_diff`)). Table 1 contains a selection of tasks that PAIGOS can process, currently.

Tasks can be *internal tasks* that are of potential interest for system components. *Public tasks*. need to be described sufficiently precise to enable a communication between different components, services, and systems. Their description contains the following information:

- the identifier of the task
- the number of concepts the task can be applied to. A task can either be applied to a single concept (cardinality 1) or multiple concepts (cardinality n)
- the type of learning object (as defined in the OIO) that the task can be applied to
- the type of course to expect as a result. Possible values are either `course` in case a complete course is generated or `section` in case a single section is returned. Even in case the course generator selects only a single learning object, the resource is included in a section.
- an optional element `condition` that is evaluated in order to determine whether a task can be achieved. An example is ACTIVEMATH's item menu

```
<tasks>
   <task>
      <pedObj id="trainWithSingleExercise!"/>
      <contentIDs cardinality="1"/>
      <applicableOn type="fundamental"/>
      <result type="section"/>
      <condition>(class Exercise)(relation isFor ?c)
                 (property hasLearningContext ?learningContext)
      </condition>
      <description>
         <text xml:lang="en">Train the concept.</text>
         <text xml:lang="de">\"Ube den Inhalt.</text>
      </description>
   </task>
   ...
</tasks>
```

Figure 5: A public task representation

for adding content. Its entries are displayed only if the corresponding tasks can be achieved. E.g., if there are no examples available for `def_slope`, then the task (`illustrate`, (`def_slope`)) cannot be achieved, so it wont be displayed.

- a concise description of the purpose that is used for display in menus. **•EdNote!**

Carsten: wofuer ist das? habe nur eine task gelassen, da wir Platz sparen mssen, erica

•EdNote! Figure 5 displays the task `trainWithSingleExercise!`. It is applicable to a single educational resource of the type `fundamental` and returns a result in case the condition holds.

3.3.2 Course Generation Interfaces and Communication

PAIGOS provides two main Web interfaces, the core interface contains the methods for the course generation and the interface that allows a client to register at PAIGOS. The core interface consists of the following methods:

1. The method `getTaskDefinitions` is used to retrieve the pedagogical tasks which the course generator can process.
2. The method `generateCourse` starts the course generation on a given task. The client can make information about the learner available either through a pointer to his/her student model or by a list of property-value pairs. .

In order to achieve interoperability, the Web interface returns an IMS Manifest, consisting of references to LOs. The internal interface returns a table-of-contents like structure whose leaves are references or a special kind of tasks called dynamic tasks which are instatiated only later.

The interface for repository registration consists of the following methods:

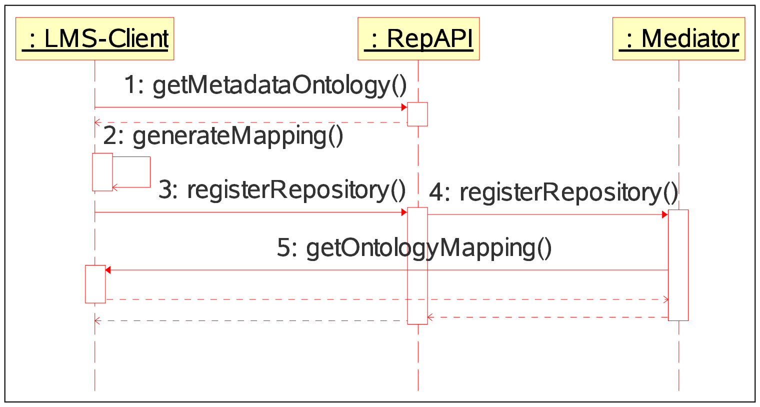

Figure 6: Sequence diagram of repository registration

1. The method `getMetadataOntology` informs the client about the metadata structure used in Paigos. It returns the ontology of instructional objects.
2. The method `registerRepository` registers the repository that the client wants the course generator to use. The client has to provide the name and the location (URL) of the repository. Additional parameters include the ontology that describes the metadata structure used in the repository and the mapping of the OIO onto the repository ontology.
3. The method `unregisterRepository` cancels the registration of a repository.

The interface *ResourceQuery* is used to query the repository about properties of learning objects. The interface consists of the following methods:

1. `queryClass` returns the classes a specified resource belongs to
2. `queryRelation` returns the set of identifiers of those learning objects the given resource is related to via the specified relation
3. `queryProperty` returns the set of property-value pairs the specified resource has.

The *LearnerPropertyAPI* makes the learners' properties accessible to Paigos in case the client has a learner model and wants the course generator to use it. In the current version of Paigos, this interface is not yet implemented. It would require a mediator architecture similar to the one used in ActiveMath for repository integration [Kärger et al.(2006)].
The result of the course generation is a structured sequence of learning objects represented in an IMS Manifest[IMS(2003)]. Since the result does not contain the resources but only references, it is not an IMS CP.

A repository is registered in the following way (for a sequence diagram illustrating the registration, see Figure 6): in a first step, the client (LMS-Client in the figure) retrieves the metadata ontology used in Paigos. The ontology is then used to generate a mapping between the OIO and the ontology representing the client metadata (Step 2) (the currently existing mappings were manually authored). Then, the repository is registered using the method `registerRepository` (Step 3). The repository is added to the list of available repositories and made known to the mediator (a component of Paigos that allows the integration of third-party repositories) (Step 4). Subsequently, the mediator fetches the ontology mapping from the client and automatically generates a wrapper for querying the `contentAPI` of the client.

A client starts the course generation using the service method `generateCourse`. In a first step, Paigos checks whether the task is valid. If so, the course is generated. During the generation process, Paigos sends queries to the mediator, which passes the queries to the repository. After the course is generated, the `omgroup` (the element `OMDoc` uses for grouping elements) generated by Paigos is transformed into an ims Manifest and sent to the client.

4 Presentation and Management of Mathematical Expressions

For different users, mathematical formulæ can be presented in different formats (for print or browser) and forms – depending on the user's (cultural) context and preferences. Moreover, the diversity of the rendering forms also builds on the diversity of the notations in mathematical practice, e.g., the fact that $\sin^2 x$ is written without bracket while $\sin^2(x+y)$ is written with brackets even though the mathematical symbol is the same.

Most solutions for browser rendering mathematics focus on a single presentation language which can be rendered in multiple browsers. For instance, JS-math[2] or Wikipedia's texvc[3] use a subset of LaTeX to allow for authoring of mathematical formulæ with presentation markup, i.e., markup usable only for rendering.

This does not, however, solve the presentation problem for student interactions and search as needed in an e-Learning system. In order not to confuse the student, to avoid cognitive overload, and to support smooth interaction with content and tools the presentation of mathematical expressions should be the same in all tools of his/her learning experience. For instance, when the learner uses a curve plotter, the lexicon/search tool, the input editor, a CAS service, he/she should view the same presentation of a symbol in any application. At first, this seems to be trivial but it is not:

1. interactions occur in interactive exercises for which the student's input is evaluated. Interactions also occur with (GUIs of) interactive tools such as computer algebra systems or function plotter. In this case, the semantic and computable nature of the mathematical object is required for consistency. Hence, the presentation in any of the tools cannot be hard coded but needs to be generated. The generation of a

[2]JSmath is a javascript library that renders TeX within the browser, see `http://www.math.union.edu/~dpvc/jsMath/`

[3]texvc is an add-on to mediawiki explained at `http://en.wikipedia.org/wiki/Texvc`.

presentation is also required because of the need of cultural adaptation which requires to use culture-specific presentations for a number of symbols and expressions.

2. search for mathematical formulæ needs to be independent of the actual rendering and should exclude mismatches such as $x+y^2$ when the user queried for x^2.

ActiveMath responds to the need to render formulæ consistently in the content as well as in interactions and to search semantically by processing formulæ in `OpenMath` as the semantic basis for presentation and management (search, input, copy and paste, etc.) of mathematical expressions.

4.1 Adaptive Rendering of Formulæ

Following the common web practice, all interactions in ActiveMath occur in a web-browser and applets. The browser's interactions with the web-server involve the generation of a presentation code (refer to Figure 1). As much as possible, ActiveMath uses its generic presentation architecture to produce the rendering of mathematical formulæ based on their semantic representation. Rendering of formulæ is part of the presentation process of ActiveMath which aims at delivering browser code of the content. This delivery depends on the context and preferences of the user which includes the following dimensions:

1. the format of delivery, which is mostly a choice of the user (currently HtML +css, TeX/PDF, and xHtML+MathML are supported)
2. the language of the user, which impacts the notations
3. the educational context and field of study
4. the course that is currently delivered.

The delivery converts `OMDoc` items into chunks of browser code based on the format, language, and notation. XSLT transformations are used to this end. The XSLT transformations are partially generated by a set of notations which associates `OpenMath` prototypes (expressions with variables as place holders) to a presentation template. The resulting adapted rendering of mathematical content is in line with a user's cultural customs while at the same time it keeps its meaning through the underlying `OpenMath` expressions.

4.2 Input of Mathematical Formulæ

An interaction for which `OpenMath` is crucial too, is the input of formulæ. In ActiveMath mathematical formulæ can be input in three ways:

Input Editor: The input editor of ActiveMath is palette-based and can be used in different platforms. It is easily accessible to a novice user for the input of symbols. It is implemented as a Java applet which internally edits an `OpenMath` expression. Its palettes are configurable by a skilled author. Its rules for transforming `OpenMath` expressions to a rendering code employ internal rules and notations central to ActiveMath – thus achieving consistency for a student.

Textfields: Because not all students want to work with such an input editor ActiveMath enables linear input syntax as well. Its syntax resembles that of the Maple and its output is `OpenMath`.

Linear Input for Authors: In the authoring environment mathematical formulæ are input with a linear syntax (OQMath), which is configurable by notation files.

These methods can be complemented by a copy-and-paste facility: a feature of ActiveMath's presentation is a reference to a URL at which the `OpenMath` term is available. The paste of the URL representing this term is interpreted by the input editor and other recipients (linear input, function plotter, etc.) as a request to fetch and insert the `OpenMath` term.

5 Conclusion

The article described several Semantic Web features of the e-Learning platform ActiveMath. The backbone of many of those features is the semantic knowledge representation for mathematics, `OMDoc`, and its ActiveMath metadata extensions. This representation is processed by a number of components of the platform whose architecture and communication is conveyed at a high level.

We explain relevant features of web services and components used for adaptive course generation, for the diagnosis of student input in exercises, as well as the presentation and management of mathematical expressions.

5.1 Future Work

The oio ontology has been adopted and extended by other groups. We hope this will also happen to the interoperable services which are currently used by ActiveMath. Currently, the ActiveMath group is in the process of reusing learning material originally devised for other learning environments. For this purpose, however, mathematical semantics and metadata have to be added to the content.

Currently, ActiveMath can exchange basic student profile information with other applications such as Moodle but does not (yet) exchange detailed student model information. This is a future goal.

Paigos was successfully used by the two third-party systems MathCoach (a learning tool for statistics [Grabowski et al.(2005)]), and Teal (work flow embedded e-learning at the workplace, [Rostanin et al.(2006)]). Future work on Paigos is necessary to realize a mediator-like architecture for the generic integration of/communication with student models of other applications.

The search tool of ActiveMath has almost been neglected in this article even though it searches semantically by matching formulæ and their `OpenMath` trees and its search for LOs can integrate metadata. Its match of `OpenMath` trees is exact, i.e., no equivalent formulation is returned yet. That is, for $x + y$ only $x + y$ would be returned but not $y + x$. Normalization is a first step to cope with various equivalent encodings. Future work will deal with more fuzzy search.

5.2 Acknowledgement

This publication has been supported by projects LeActiveMath () funded by the EU and ATuF (ME 1136/5-1) funded by the German National Science Foundation (DFG). The authors are solely responsible for its content.

The authors wish to thank Tianxiang Lu for the implementation of the work described in Section 3.3.

References

[Anderson et al.(2001)] L.W. Anderson, D.R. Krathwohl, P.W. Airasian, K.A. Cruikshank, R.E. Mayer, P.R. Pintrich, and M.C. Wittrock. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxomnomy of Edicational Objectives.* Longman, New York, 2001.

[Bloom(1956)] B.S. Bloom, editor. *Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain.* Longmans, Green, New York, Toronto, 1956.

[Brooks et al.(2006)] C. Brooks, J. Greer, E. Melis, and C. Ullrich. Combining its and elearning technologies: Opportunities and challenges. In K.D. Ashley M. Ikeda and T-W. Chan, editors, *Intelligent Tutoring Systems (ITS-06)*, volume 4053 of *LNCS*, pages 278–287, Jhongli, Taiwan, 2006. Springer-Verlag.

[Conlan et al.(2002)] O. Conlan, V. Wade, C. Bruen, and M. Gargan. Multi-model, metadata driven approach to adaptive hypermedia services for personalized elearning. In P. De Bra, P. Brusilovsky, and R. Conejo, editors, *Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 2347 of *LNCS*, pages 100–111. Springer-Verlag, 2002.

[Dudev, González Palomo(2007)] M. Dudev, A. González Palomo, Generating Parametrized Exercises, Student Project at the University of Saarland, March 2007, PDF `http://www.matracas.org/escritos/edtech_report.pdf`

[Faulhaber and Melis(2008)] A. Faulhaber and E. Melis. An efficient student model based on student performance and metadata. In N. Fakotakis M. Ghallab, C.D. Spyropoulos and N. Avouris, editors, *18th European Conference on Artificial Intelligence (ECAI-2008)*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 276–280. IOS Press, 2008.

[Grabowski et al.(2005)] B. Grabowski, S. Gäng, J. Herter, and T. Köppen. MathCoach und LaplaceSkript: Ein programmierbarer interaktiver Mathematiktutor mit XML-basierter Skriptsprache. In Klaus P. Jantke, Klaus-Peter Fähnrich, and Wolfgang S. Wittig, editors, *Leipziger Informatik-Tage*, volume 72 of *LNI*, pages 211–218. GI, 2005.

[Heeren at al.(2008)] B. Heeren, J. Jeuring, A. van Leeuwen, and A. Gerdes. Specifying Strategies for Exercises. In Serge Autexier, John Campbell, Julio Rubio, Volker Sorge, Masakazu Suzuki, Freek Wiedijk, editors, AISC/Calculemus/MKM 2008, LNAI 5144, pages 430 – 445, 2007, Springer-Verlag. 2008.

[IMS(2003)] IMS Global Learning Consortium. IMS content packaging information model, June 2003.

[Karampiperis and Sampson(2005)] P. Karampiperis and D. Sampson. Adaptive learning resources sequencing in educational hypermedia systems. *Educational Technology & Society*, 8(4):128–147, 2005.

[Kärger et al.(2006)] P. Kärger, C. Ullrich, and E. Melis. Integrating learning object repositories using a mediator architecture. In Wolfgang Nejdl and Klaus Tochtermann, editors, *Proceedings of ECTEL'06*, volume 4227, pages 185–197, Heraklion, Greece, October 2006. Springer-Verlag. ISBN 9783540457770.

[Keenoy et al.(2005)] K. Keenoy, A. Poulovassilis, G. Papamarkos, P.T. Wood, V. Christophides, A. Maganaraki, M. Stratakis, P. Rigaux, and N. Spyratos. Adaptive personalisation in self e-learning networks. In *Proceedings of First International Kaleidoscope Learning Grid SIG Workshop on Distributed e-Learning Environments*, Napoly, Italy, 2005.

[Knight et al.(2006)] C. Knight, D. Gašević, and G. Richards. An ontology-based framework for bridging learning design and learning content. *Educational Technology and Society*, 9(1):23–37, 2006.

[Kohlhase(2006)] M. Kohlhase OMDoc: An Open Markup Format for Mathematical Documents (version 1.2). LNCS 4180, Springer-Verlag, Berlin, 2006.

[Krämer(2005)] B.J. Krämer. Reusable learning objects: Let's give it another trial. Forschungsberichte des Fachbereichs Elektrotechnik ISSN 0945-0130, Fernuniversität Hagen, 2005.

[Maple] `http://www.maplesoft.com`

[Mathematica] `http://www.wolfram.com`

[Maxima] `http://maxima.sourceforge.net`

[Melis et al.(2006)] E. Melis, G. Goguadze, M. Homik, P. Libbrecht, C. Ullrich, and S. Winterstein. Semantic-Aware Components and Services of ActiveMath. *British Journal of Educational Technology*, 37(3):405–423, may 2006.

[Melis et al.(2008)] E. Melis, A. Faulhaber, A. Eichelmann, and S. Narciss. Interoperable competencies characterizing learning objects. In E. Aimeur B.Woolf and R. Nkambou, editors, *Proceedings of the International Conference on Intelligent Tutoring Systems, ITS-2008*, volume 5091 of *LNCS*, pages 416–425. Springer-Verlag, 2008.

[Merceron et al.(2004)] A. Merceron, C. Oliveira, M. Scholl, and C. Ullrich. Mining for Content Re-Use and Exchange – Solutions and Problems. In *Poster Proceedings of the 3rd International Semantic Web Conference, ISWC2004*, pages 39–40, Hiroshima, Japan, November 2004.

[MONET(2003)] MONET Architecture Overview, The MONET Consortium, Deliverable D04, March, 2003

[OECD(2004)] OECD, editor. *Learning for Tomorrows World – First Results from PISA 2003*. Organization for Economic Co-operation and Development (OECD) Publishing, 2004.

[Rostanin et al.(2006)] O. Rostanin, C. Ullrich, H. Holz, and S. Song. Project teal: Add adaptive e-learning to your workflows. In Klaus Tochtermann and Hermann Maurer, editors, *Proceedings: I-KNOW'06, 6th International Conference on Knowledge Management*, pages 395–402, Graz, Austria, Sep 2006.

[Sancho et al.(2005)] Pilar Sancho, Iván Martínez, and Baltasar Fernández-Manjón. Semantic web technologies applied to e-learning personalization in e-aula. *Journal of Universal Computer Science*, 11(9):1470–1481, 2005.

[IEEE LTSC (2002)] IEEE Learning Technology Standards Committee. 1484.12.1-2002 IEEE standard for Learning Object Metadata, 2002.

[Ullrich(2005)] C. Ullrich. The learning-resource-type is dead, long live the learning-resource-type! *Learning Objects and Learning Designs*, 1(1):7–15, 2005.

[Ullrich(2008)] C. Ullrich. *Pedagogically Founded Courseware Generation for Web-Based Learning – An HTN-Planning-Based Approach Implemented in PAIGOS*. Number 5260 in Lecture Notes in Artificial Intelligence. Springer, 2008. ISBN 978-3-540-88213-8.

[Van Marcke(1998)] Kris Van Marcke. GTE: An epistemological approach to instructional modelling. *Instructional Science*, 26:147–191, 1998.

[1] `http://www.wiris.com`

[2] `http://yacas.sourceforge.net`

[Zimmer, Autexier(2006)] J. Zimmer and S. Autexier. The MathServe System for Semantic Web Reasoning Services, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130, pages 140–144, Springer Verlag, August 2006.

[Zinn(2006)] C. Zinn. Supporting tutorial feedback to student help requests and errors in symbolic differentiation. In K. Ashley M. Ikeda, editor, *Proceedings of Intelligent Tutoring Systems 8th. International Conference ITS-2006*, volume LNCS 4053 of *Lecture Notes in Computer Science*, pages 349–359. Springer-Verlag, June 2006.

[OpenMath] O.Caprotti, D.P.Carlisle and A.M. Cohen. The OpenMath Standard. The OpenMath Consortium, 2002.

[OpenMath-CDs] J.Davenport et al. OpenMath Core Content Dictionariers. April, 2004, `http://www.openmath.org/`.