

ACTIVEMATH – a Learning Platform With Semantic Web Features

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

March 13, 2009

Abstract

ACTIVEMATH is an intelligent e-Learning system that exhibits some 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. In this paper, we describe features which have not been presented at all or only superficially in previous publications.

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

1 Introduction

ACTIVEMATH was one of the first systems to seriously address 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 expert/domain model, a student model, and pedagogical model/modules including course generator, tutorial strategies, and feedback generators.

Different from most ITS it encodes the domain model 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 is evolving and changing over time. Hence, ACTIVEMATH has to take care of those changes.

What is also rather untypical for an ITS are the advanced features that make it a Semantic Web application, e.g., semantic search, truly semantic markup and reuse of content, generation of web presentations from the representation of content, interoperable content and components, distributed architecture, asynchronous event framework, etc. Hence, ACTIVEMATH is also a workbench for

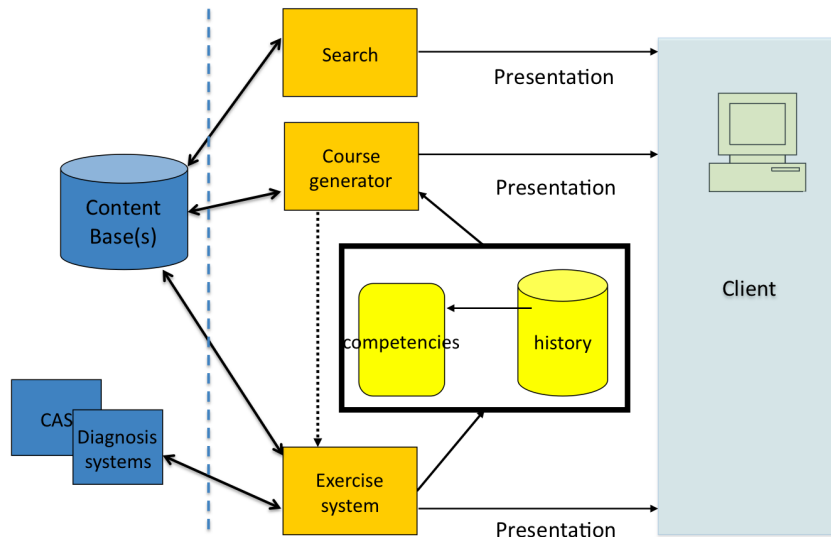


Figure 1: Coarse architecture of ActiveMath with services

studying benefits of combining ITS and semantic e-learning technologies as suggested in [Brooks et al.(2006)].

In the following, we describe web- and semantic-web-features which have not been included at all or only superficially in previous articles.

2 Preliminaries about ACTIVEMATH

In order to provide a rather self-contained chapter, we start with briefly summarizing some of ACTIVEMATH’s features which were already published in previous publications, mainly in [Melis et al.(2006)].

Figure 1 coarsely depicts the server- and client-side of the ACTIVEMATH platform and its web-service communications with external servers (left-hand-side). One central component of ACTIVEMATH is its course(ware) generator, called PAIGOS [Ullrich(2008)]. A course generator 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. In contrast to current course generators [Conlan et al.(2002), Keenoy et al.(2005), Karampiperis and Sampson(2005)], PAIGOS is based on an extensive model of expert teaching knowledge – about 300 “rules” define how to assemble different types of courses. PAIGOS also functions as a service and can be accessed by other learning environments. Section 2.2 describes the requirements on knowledge representation to enable such a service, and Section 3.3 describes the service in detail.

2.1 Semantic Knowledge Representation

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

The `OMDoc` format itself is using `OpenMath` [`OpenMath`] as embedded format for representing mathematical formulæ. `OpenMath` is a well-established standard for representing mathematical formulæ. `OpenMath` defines its semantic as a mathematical language by the usage of so-called content dictionaries as explained in [`OpenMath`]. The content dictionaries contain symbol declarations which provide central hooks to which symbol occurrences point to when using the `OMS` element. The symbol declarations are complemented by a description in regular English and by formal properties which are mathematical statements that should stay true for the symbol to be interoperable. The content dictionaries define what is agreed upon when emitting and processing `OpenMath` expressions. This enables a semantic evaluation and a search for mathematical formulæ.

`OMDoc` extends this format in two directions: all textual fragments can be made in multiple languages and be interleaved with formulæ and it extends the structure options by a grouping construct called `theory` which models the concept of a formal mathematical theory allows a rich management of namespace by the usage of imports. Referencing the structure, e.g. `theory`, to which a symbol belongs adds semantical information to mathematical expressions (e.g., whether `+` is plus for real numbers or for matrices) which provides a basis for their semantic evaluation. This is important for external mathematical reasoning services used for diagnosing user input. Not for all services a disambiguated semantics of all symbols is needed. Some computer algebra systems (CAS) use symbol overloading for the cases in which a disambiguation of symbols is possible. For instance, symbol `'+'` might be used both for adding integers and matrices and some CAS can disambiguate the semantics of the operation by analyzing the arguments. Such overloading is used to make human interaction with the CAS easier. But domain reasoning services, which are supposed to be only used as back end engines for semantic evaluation cannot resolve any ambiguities and, therefore, need formulæ with proper semantics.

`ACTIVEMATH` can communicate with CASs and domain reasoners which have so-called phrasebooks translating the `OpenMath` input into their mathematical representation and vice versa translating computation result back to `OpenMath` for `ACTIVEMATH`.

2.2 Ontologies

ACTIVEMATH uses two different ontologies. One ontology is represented within pure OMDoc and describes the subject domain from a mathematical point of view. The second ontology contains the pedagogical information and is represented in our extension of OMDoc. It defines types (or classes) of learning objects according to their instructional function and properties of LOs. It is independent of the specific subject domain.

We had to develop such an ontology because existing learning object metadata standards such as LOM [IEEE Learning Technology Standards Committee(2002)] failed to describe learning objects sufficiently precise for intelligent components to integrate them automatically into the students' learning work flow. ACTIVEMATH's *ontology of instructional objects* (OIO) [Ullrich(2005)] contains specifically this previously missing information. Its classes are shown in Figure 2.

The ontology enables several of ACTIVEMATH' advanced pedagogical features. It allows to define the course generation knowledge such that it is independent of the specific mathematical domain. The "rules" can be applied to teach differentiation as well as group theory. 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). Furthermore, the ontology facilitates the process of making third-party repositories available to the course generator. On the one hand, this process helps to assemble a course from resources of different repositories. On the other hand, the ontology helps to enable the course generator to provide its functionality as a service to systems that plug in their repositories (described in Section 3.3).

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)].

2.3 Metadata

Metadata used by ACTIVEMATH can be divided in three main categories: general administrative metadata, mathematical metadata, and educational metadata.

For general annotations of LOs such as **title** of the item, **date** of its creation or modification, names of **authors**, 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 Creative Commons metadata.

Mathematical metadata define mathematical types of LOs 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)

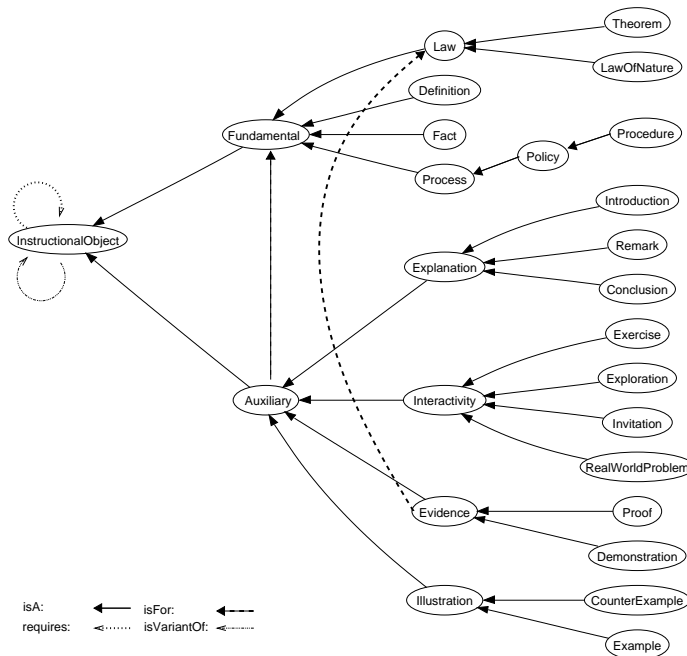


Figure 2: Overview of the Ontology of Instructional Objects

the **for** relation indicating that the current LO relates to a concept/symbol, e.g., to define, explain, illustrate, train a concept (a symbol, theorem, or definition).

Educational 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 ACTIVE MATH to act intelligently and to model the student.

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

3 Web-Services and Components of ACTIVE MATH

3.1 Diagnostic Services

ACTIVE MATH 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. ACTIVE MATH

can query two kind of services for diagnosing the student's input, generic computer algebra systems (CASs) and specifically designed domain reasoner services which provide more human-like reasoning and possibly incorrect rules in different mathematical domains.

The semantic `OpenMath` markup for mathematical formulae and a generic format for queries to the diagnostic services (see below) support the interoperability of different CASs and reasoners when serving complex domains. For each diagnostic service, the `OpenMath` formulae are translated to and from the syntax of the service via an `OpenMath` phrasebook. Due to the fact, that the `OpenMath` format represents semantics of mathematical formulae, such phrasebooks can always be implemented. Currently, `ACTIVEMATH` integrates and communicates with the following CASs: `YACAS` [5], `Maxima` [3], and `WIRIS` [4]; phrasebooks for `Maple` [1] and `Mathematica` [2] are available too.

Figure 3 shows different ways of connecting to CASs that we realized in our framework dependent on a CAS's existing implementations: the `WIRIS` CAS server is connected to `ACTIVEMATH` via XML-RPC and contains an internal `OpenMath` phrasebook. The `YACAS` server that was developed in `ACTIVEMATH` group 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.

The most generic CAS connected to `ACTIVEMATH` is currently `YACAS`, since it is modular and easily extensible. New domains can be attached to `YACAS` by exchanging or extending the current 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. Moreover, 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.

CASs are very efficient and fast in providing diagnoses needed for the generation of a flag feedback (correct/incorrect) as well as for the final correct solution for the 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 numbers and intervals, but also any set of mathematical functions and their superpositions. Since the solution of each step of a problem is represented as a mathematical expression, for each randomized exercise answers/input from the student can be diagnosed as correct or incorrect by a CAS. For more information about the randomizer component see [Dudev, González Palomo(2007)].

More detailed diagnoses can be obtained when a domain reasoner is available for the mathematical domain of the exercise. A domain reasoner can send responses to queries which are used to generate common types of hints for the learner such as

- next step hint
- 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 (partial) solution of the student and diagnose his/her errors. Another domain reasoner connected to ACTIVEMATH is MathCoach [Grabowski et al.(2005)], which is stateless and cannot trace a student's solution.

Another interesting example is the series of domain reasoners implemented withing an 'Intelligent Feedback Project' at the Open Universiteit in Nederland [Heeren et al.(2008)]. Several domain reasoners have been programmed using programming language Haskell¹ and offered as reasoning web services answering queries similar to one we define in ACTIVEMATH (see section 3.1.2. Among areas/tasks covered by these domain reasoners are, e.g. rewriting logical expressions into disjunctive normal form, solving linear equations, reducing matrices to echelon normal form, and basic operations on fractions. These services offer incremental reasoning, also checking for syntactical errors as well as matching common semantic errors using buggy rules.

3.1.1 Service Query Architecture

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

ACTIVEMATH implements a novel service architecture for the diagnosis of student's actions in mathematical problem solving. The diagnosis task imposes some requirements upon such services, which we describe below. In ACTIVEMATH, 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. For instance, a domain reasoner for symbolic differentiation is only queried for (sub-)problems in symbolic differentiation. The subscribed mathematical services themselves can also send a query back to the **Query Broker**. 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 arithmetic domain reasoner.

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*.

¹see <http://www.haskell.org/>

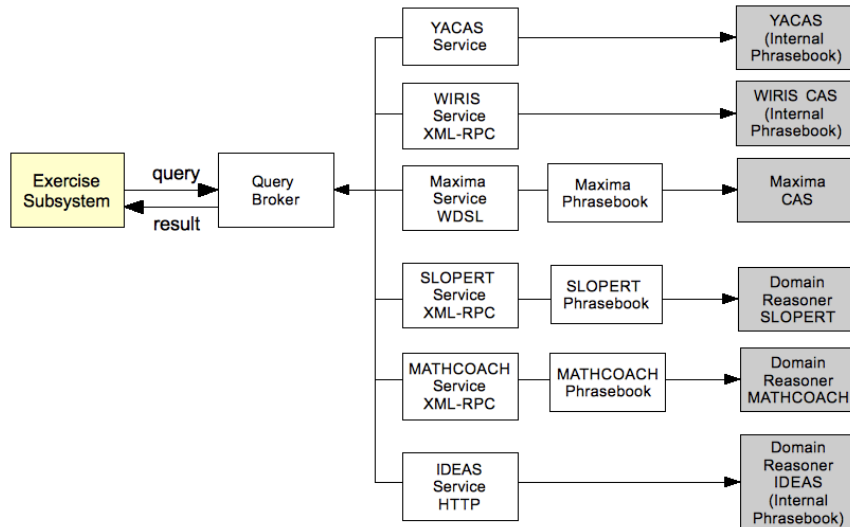


Figure 3: Diagnosis framework architecture

A context defines (sub-)sets of rules and functions that a domain reasoner or a CAS is allowed to use for its diagnosis. The background for this restriction is that the student's learning situation determines which 'rules' and functions he/she is capable to employ. Hence, the diagnosis need to simulate this rather than accepting a student input saying 'solve'.

Consider the following example: The task of the student is to differentiate a 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 and feedback generation we defined the format for queries in which a query to the domain reasoner service consist of:

- **action** of the query, e.g. `getResults`, `getUserSolutionPaths`
- **(list of) input expressions** to be evaluated or compared with each other, e.g., evaluating or simplifying task, comparing a user answer with correct answer
- **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 `ACTIVE-MATH`:

- `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, e1,e2, 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, e1, e2, 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, e1, e2, 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, e1, e2, C)` - returns 'true' if the expression e_2 is closer than e_1 to the actual solution in the context C ,

Several simple contexts can form a composite context by concatenating sets of their rules. Consider a following example query to a domain reasoner:

Example query: *Calculate 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.*

Using our query format we can formalize the example query as follows:

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

where C is the composite context consisting of arithmetical context and differential rules without product rule.

3.2 Student Model

Interoperability of a student model framework requires its ability to adopt different frameworks for competencies/skills which are used in different intelligent educational softwares. Moreover, as a web-based system that relies on content

produced by a community of authors ACTIVE MATH needs to adapt to the potentially changing structure of the domain model and, hence, of the competency model as well. Therefore, the structure of ACTIVE MATH' the student model is generated from the metadata including relations available in the content representation.

In order to cope with the potentially modified (implicit) content/domain ontology the *semantics-aware student model* (SLM) dynamically extracts relations and metadata from the ontology and makes use of their semantics. These, together with data from student interactions, enable SLM to estimate students' competencies. Moreover, ACTIVE MATH' student model is flexible enough to act upon web contents using different competency frameworks and it can handle the semantics for a number of competencies. Currently, it can choose between the competency taxonomies used in PISA [OECD(2004)], in Bloom's taxonomy [Bloom(1956)], in a comprehensive up-to-date taxonomy as described in [Anderson et al.(2001), Melis et al.(2008)], and in an extension of the latter.

In the content, the metadata related to competencies refer to a taxonomy chosen by the author. For instance, the PISA specification for mathematics 'competencies' includes

1. think
2. argue
3. model
4. solve
5. represent
6. language
7. tools.

In the following, 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)].

The generation of the student model is data-driven. The structure of SLM consists of nodes, each for a single rule/concept to estimate competencies for. SLM automatically creates a node for each concept/rule 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 .

Inter-node relations are dynamically extracted from the content metadata, i.e., the implicit domain ontology. Most important is the **prerequisite** relationship.

For each competency, beliefs about the competency values (represented by the nodes) are computed separately from recent evidences. In this computation, relations between exercises and concepts/rules as well as exercise competency metadata determine to which nodes evidences are attributed. Propagation along the **prerequisite** relations adds additional information for the competency estimation. Again, see Figure 4 for an illustration.

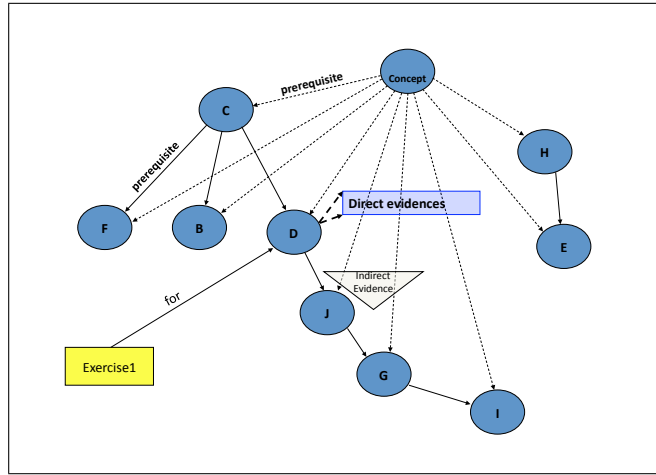


Figure 4: Structure of the student model and its updating links

3.3 Course Generation Service

The course generator of ACTIVEMATH (PAIGOS) is designed as an independent component whose services can be accessed by other learning environments, too. This requires that PAIGOS and the third party learning environments share a common understanding of the type of courses that are to be generated. We describe our semantic representation of scenarios in Section 3.3.1. Then, in Section 3.3.2, we describe the interface and the process flow between PAIGOS and its clients. The specific functionality of the interface was informed by a survey among potential clients (the results are reported in detail in [Lu(2006), Ullrich(2008)]).

3.3.1 Pedagogical Tasks

The OIO enables reasoning about learning objects. However, pedagogically sensible course generation requires reasoning that takes learning scenarios into account, that is, the context of a learner who studies the content. 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 different purposes require different course of actions. For instance, a course for preparing an exam should consist of different learning objects than a course that contains 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*

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

Table 1: A selection of tasks used in PAIGOS

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 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. We can now write the scenario for a learner who wants to discover and understand these two concepts as $t = (\text{discover}, (\text{def_slope}, \text{def_diff}))$.

Table 1 contains a selection of tasks that PAIGOS can process. The table shows that tasks exist on different levels of abstraction: the highest-level tasks result in complete courses (the first four tasks in Table 1); however, tasks can also result in the selection of a single learning object.

Tasks can be “internal” tasks, used for internal course generation purposes only, or tasks that are of potential interest for other services. The second category of tasks is called *public tasks*. Public tasks need to be described sufficiently precise in order to enable a communication between different components, services and systems. The description designed for PAIGOS 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. In some situations, a service only needs to know whether a task can be achieved but not by which learning objects. An example is ACTIVE MATH's item menu that allows the learner to request additional content. Menu entries are displayed only if the corresponding tasks can be achieved. For instance if there are no examples available for `def_slope`, then the task (`illustrate, (def_slope)`) cannot be achieved.
- a concise natural language description of the purpose that is used for display in menus.

Figure 5 contains a selection of tasks. In the figure, all keywords in the `condition` element that start with ? are variables which are instantiated by the corresponding value at the time the condition is evaluated. The top element in Figure 5 describes the pedagogical task `discover`. It is applicable to several educational resources of type `fundamental`. The bottom element specifies 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 Web Interfaces

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

- The method `getTaskDefinitions` is used to retrieve the pedagogical tasks which the course generator can process.
- The method `generateCourse` starts the course generation on a given task. The client can make information about the learner available in two ways: if the learner model contains information about the specific learner, then the client passes the respective learner identifier as a parameter. In case no learner model exists, a client gives a list of property-value pairs that is used by PAIGOS to construct a temporary "learner model". The course generator performs the planning in the same way as with a real learner

```

<tasks>
  <task>
    <pedObj id="discover"/>
    <contentIDs cardinality="n"/>
    <applicableOn type="fundamental"/>
    <result type="course"/>
    <condition></condition>
    <description>
      <text xml:lang="en">Generate a book that helps a learner to
        understand the selected topics in depth.</text>
      <text xml:lang="de">Erstelle ein Buch das hilft die
        ausgew\ahlten Begriffe grundlegend zu verstehen</text>
    </description>
  </task>
  <task>
    <pedObj id="illustrateWithSingleExample!"/>
    <contentIDs cardinality="1"/>
    <condition>(class Example)(relation isFor ?c)
      (property hasLearningContext ?learningContext)
    </condition>
    <applicableOn type="fundamental"/>
    <result type="section"/>
    <description>
      <text xml:lang="en">Illustrate the concept.</text>
      <text xml:lang="de">Veranschauliche den Inhalt.</text>
    </description>
  </task>
  <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 selection of public task descriptions

model, however its access of learner properties is diverted by PAIGOS and answered using the map. Properties not contained in the map are answered with a default value.

The result of the course generation is a structured sequence of learning objects represented in an IMS Manifest[IMS Global Learning Consortium(2003)]. Since the returned result does not contain the resources but only references, it is not an IMS CP.

The interface for repository registration consists of the following methods:

- The method `getMetadataOntology` informs the client about the metadata structure used in PAIGOS. It returns the ontology of instructional objects.
- 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.
- The method `unregisterRepository` cancels the registration of a repository.

A client that wants to use PAIGOS needs to specify information about the learning objects as well as about the learner (if available).

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

- `queryClass` returns the classes a specified resource belongs to
- `queryRelation` returns the set of identifiers of those learning objects the given resource is related to via the specified relation
- `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)].

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 (i. e., the OIO). 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,

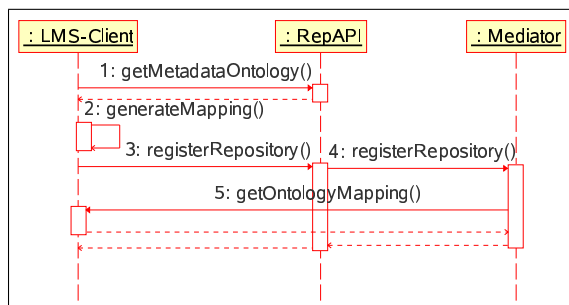


Figure 6: A sequence diagram illustrating the repository registration

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`. The argument of the method consist of a task. In a first step, PAIGOS checks whether the task is valid. If so, the course is generated by the course generator. During the generation process, PAIGOS sends queries to the mediator, which passes the queries to the repository. The results are cached in order to speed-up the course generation process. 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 Consistent Presentation and Management of Mathematical Expressions

As largely explained in §2.1, `OpenMath` is the semantic representation of mathematics in `ACTIVEMATH` on which presentation and management of mathematical expressions is based.

For different users, mathematical formulæ can be visualized in `ACTIVEMATH` in different 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.

In order not to confuse the student, 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. Therefore, the presentation in any of the tools cannot be hard coded but need to be generated. The generation of a presentation is also required because of the need of cultural adaptation which requires to use culture-specific presentations for a number of symbols/expressions.

Following the common web practice, all interactions in `ACTIVEMATH` are

done in a web-browser and applets. The browser's interactions with the web-server involve the generation of a presentation code (see again Figure 1). As much as possible, `ACTIVEMATH` uses its generic rendering architecture to produce the rendering of mathematical formulæ based on their semantic representation.

Most other web-solutions for 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 eLearning system. In order for learners to smoothly interact with content and tools and to avoid a cognitive overload it is important that the appearance/rendering of mathematical formulæ in *all* user interfaces and applications is the same. At first this seems to be trivial but it is not:

- 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.
- 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 all formulæ in `OpenMath` and by using its presentation engine as much as possible. An interaction for which `OpenMath` is crucial is the input of formulæ.

4.1 Input of Mathematical Formulæ

Mathematical formulæ can be input in three ways:

Input Editor: The input editor of `ACTIVEMATH` is palette-based and can be use in different platforms. It is easily accessible to a novice user for the input of basic 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.

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>.

Textfields: Because not all students want to work with such an input editor ACTIVE-MATH enables linear input syntax as well, a syntax resembling that of the Maple. Again, the output is `OpenMath`.

Linear Input for Authors: Within the authoring environment [Libbrecht and Gross(2006)], the input of mathematical formulæ is made along a linear syntax which is configurable by notation files.

All of these methods can be complemented by a copy-and-paste facility: a feature of the (added value) presentation service is to make an `OpenMath` term available at an URL. The drop of the URL representing this term is interpreted by the input editor and other recipients (linear input, function plotter, etc) as the wish to fetch and insert the underlying `OpenMath` expression.

4.2 Adaptive Rendering of Formulæ

Rendering of formulæ is part of the rendering/presentation process of ACTIVE-MATH which aims at delivering browser code for the content. This delivery depends on the context and preferences of the user which includes the following dimensions:

- the format of delivery, which is mostly a choice of the user (currently HTML +CSS, T_EX/PDF, and XHTML+MATHML are supported)
- the language of the user, which impacts the notations
- the educational context and field of study
- the course that is currently delivered.

The delivery converts `OMDoc` items, which contain formulæ in `OpenMath`, 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 placeholders) to a presentation template.

The resulting adaptive rendering yields a presentation of mathematical content that is in line with a user's cultural customs while at the same time it keeps its meaning through the underlying `OpenMath` expressions.

5 Conclusion

The article described several Semantic Web features of the eLearning platform ACTIVE-MATH. The backbone of many of those features is the semantic knowledge representation for mathematics `OMDoc` and its ACTIVE-MATH metadata extensions. We also describe web services used for intelligent course generation, for the diagnosis of student input in exercises, as well as the consistent presentation and management of mathematical expressions.

The OIO ontology we developed has been adopted and extended by other groups. We hope this will also happen to the interoperable services which are currently used by ACTIVE MATH. The ACTIVE MATH 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.

5.1 Future Work

Interoperable user models Currently, ACTIVE MATH 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 MATH COACH (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.

Fuzzy Semantic Mathematical Search The search tool of ACTIVE MATH 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. The match of **OpenMath** trees is exact which makes it appropriate only if no equivalent formulation should be returned. 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 less exact 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 Taxonomy of Educational 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 Mathematikutor 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 Global Learning Consortium(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. Poulouvasilis, G. Papamarkos, P.T. Wood, V. Christophides, A. Maganaraki, M. Stratakis, P. Rigaux, and N. Spyrtos. 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.
- [Krämer(2005)] B.J. Krämer. Reusable learning objects: Let’s give it another trial. Forschungsberichte des Fachbereichs Elektrotechnik ISSN 0945-0130, Fernuniversität Hagen, 2005.
- [Libbrecht and Gross(2006)] P. Libbrecht and C. Gross. Experience Report Writing LeActiveMath Calculus. *Proceedings of Mathematical Knowledge Management 2006*, J. Borwein and W. Farmer (eds), LNAI 4108, Springer-Verlag, pages: 251–265, 2006.
- [Lu(2006)] T. Lu. Kursgenerator für e-learning syteme als web-service. Master’s thesis, Hochschule fr Technik und Wirtschaft des Saarlandes, 2006.
- [Maple] <http://www.maplesoft.com>
- [Mathematica] <http://www.wolfram.com>
- [Maxima] <http://maxima.sourceforge.net>
- [Melis et al.(2006)] E. Melis, G. Gogvadze, M. Homik, P. Libbrecht, C. Ullrich, and S. Winterstein. Semantic-Aware Components and Services of Active-Math. *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 Learning Technology Standards Committee(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 Dictionaries. April, 2004, <http://www.openmath.org/>.