Authoring LeActiveMath Calculus Content

Paul Libbrecht and Christian Gross

[1] DFKI GmbH, Saarbrücken, Germany
[2] Ludwigs Maximilans Universtität, München, Germany

Abstract. Within the LeActiveMath project, a collection of OMDoc files and supporting material has been realized. This content covers the derivative side of calculus and is being used by students in the LeActiveMath learning environment. LeAM-calculus is the first collection trying to make use of most of the features of the learning environment including advanced usages of OpenMath and OMDoc. It has been written in OQMath, a readable xml-syntax.
This paper describes the tools to produce it, how they were used and combined, the resulting content and the experience gained. It argues that the declaration of new OpenMath symbols is a requirement and explains challenges of authoring semantic mathematical content. Finally, it presents the management activities to support the authoring process.

Introduction

This paper is an experience report on a field yet little experimented with: the creation of semantic mathematical documents for use in an integrated learning environment. It is organized as follows: first we describe the expected deliverable of the authoring activity, and its ingredients. Third-party tools that could answer these missions are covered. We provide a description of the tools used to fulfill the mission. The collection is, then, depicted in numbers and characteristics. Challenges met during the authoring activity are then described with a description of tools realized to (partially) answer them. A special accent is put on the management of mathematical knowledge. Finally, open challenges are described with a hint to future work.

1 The mission

Among the goals of the LeActiveMath project is to prove the learning-efficiency and acceptance of the tools developed within the research project based on usage within real educational settings. For this to happen, a large content collection was to be written. This content should showcase the usage of the advanced features of the learning environment:

– It should support the semantic mathematical approach used in the OpenMath [3] encoding which makes it possible to export formulæ to such tools as computer-algebra systems for exercise evaluation [4] or to search for them.

- It should support the macro-level semantics of mathematical documents as proposed by OMDoc and extended in the ActiveMath [11] and LeActiveMath projects where the domain knowledge is represented using competencies inspired by the Pisa study [13]
- It should provide a sufficient amount of learning content for the adaptivity to be realized. LeActiveMath's adaptivity is realized, mostly, using the tutorial component [15] which selects content elements from a wide choice using founded pedagogical strategies.
- It should make proper use of the tools developed in the LeActiveMath project such as the concept-mapping exercise and cognitive-support tool [10], the copy-and-paste facility, ...
- It should be available at least in German, English, and Spanish to demonstrate the language adaptivity of the learning environment.
- The content should be presented with a high-quality formula rendering within web-browsers using the presentation architecture and notation system [9].

The authors of the content should be experts in both pedagogy and mathematics and will be trained in the features of the learning environment and in the usage of the authoring tools to write such content. The authors are to stay in close contact with tool developers in order to ensure that their expectations, as educationalists and mathematicians, are met and in order for them to take advantage of the evolving features.

1.1 Elements of OMDoc Authoring

The content in LeActiveMath is encoded with the OMDoc language [7]: this language extends the OpenMath standard [3] by a formalism for mathematical documents. In turn this language has been enriched to support more pedagogical metadata [11]. The content to be input is made of textual or conceptual items which have a mathematical *role*, e.g. a proof, a definition, or an exercise. Each item has an identifier and can be served individually.

The items are annotated with knowledge to present it (e.g., titles, authorship), and for it to be inspected by the tutorial component or learner model (such as the educational context it is aimed at, its difficulty, or the competencies and concepts it is intended to). These annotations, grouped in an element called *metadata*, also contain relations between the items, e.g., the relation between an exercise and the concepts that are trained in this exercise, or the relation between a theorem and its proof(s) and corollaries. The input of these relations requires an author to know well the content or be able to browse through it easily.

The items unless purely conceptual will be presented to learners. Their content is written using text mixed with mathematical formulæ in OpenMath. This plain text is enriched with a small amount of markup such as the links to items or the embedding of resources such as pictures or applets. Each textual fragment is flagged with a language which makes it possible for the content's organization in items to be multilingual.

Many references are expected to be input by the author. They include explicit links on a piece of text, relations in the metadata, transitions in response to exercise evaluations, inclusion of an item in a *book*, as well as OpenMath symbol references in the form of `OMS` elements.

2 Other Approaches

In this section we describe existing approaches in the direction of authoring semantic mathematical content or content for adaptive hypermedia.

2.1 Authoring of Mathematical Semantic Content

Semantic mathematical documents to be played on the web are not, yet, commonly authored, especially with the requirements of a *macro-structure* slicing items with mathematical roles and metadata as well as that of an extensible set of mathematical symbols. To our knowledge the following tools work with the same requirements and they all work on OMDoc: STEX [8] a LaTeX-macro package which can produce OMDoc, CPoint [6] an extension to PowerPoint to support annotations of slides with OMDoc metadata, and QMath which we shall describe later. To all of these approaches we can at least state two missing wishes: error reporting about the produced OMDoc seems not possible or considered and support to suggest possible insertions while editing may be missing.

The Connexions project [5] aims at the development of an open-source commons for scholarly content. The content with formulæ is expected to be authored in XML-source form. Connexions relies on MathML-content for mathematical formulæ with little for structuring parts of the content-*modules*. Because of this, the project lacks features such as authorable notations or the context needed to provide definitions and supporting material around the introduction of a mathematical symbol as a conceptual entity.

Many other approaches exist to help in the publication of mathematical documents on the web. For most of them the ability to define items and metadata and the support for semantic mathematical objects is less important. They include LaTeX exporters, extended word-processors, mathematical assessment systems, or computer algebra systems. Only the two latter categories do export semantic mathematics and, in their case, this export is not easily extensible to support the addition of new symbols. Mathematical assessment systems, e.g. as presented at the WebALT conference[3], are still mostly centered on fixed computer algebra system whose input-syntax and formula rendering is used. An export to semantically encoded documents is rarely available.

We insist on the fact that, as has been shown in [11], the effort of authoring semantic mathematical formulæ pays on the long run. An example is the

[3] The WebALT conference took place at the beginning of the year 2006, see `http://webalt.math.helsinki.fi/webalt2006` for the programme and access to most papers.

international character of the expressions which becomes language-specific when presented (e.g. presents “ggT” or “gcd”). Such an expression authored in a non-semantic way would require different expressions for each language. Semantic formulæ are also the basis of most added value to a formula display such as the possibility to click to see definitions of a symbol or to copy-and-paste an expression.

2.2 Authoring Content for Adaptive Environments

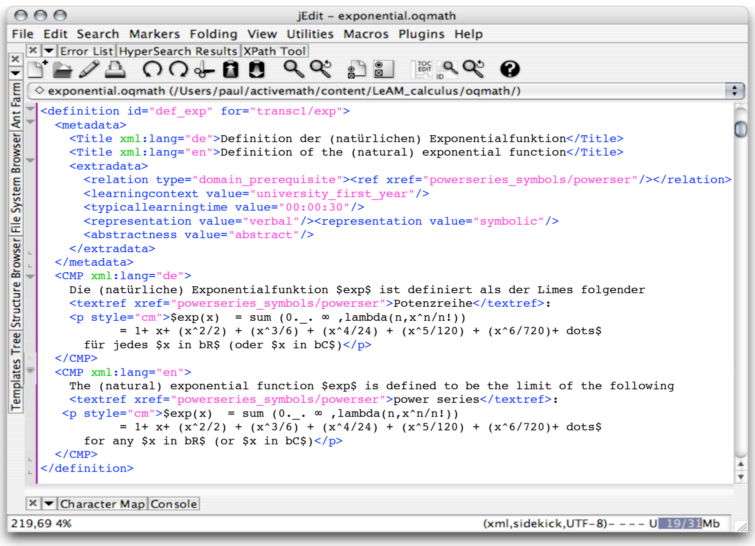
The field of authoring tools for adaptive environments, or more generally for *Advanced Technology Learning Environments* is rich and diverse. It aims at solving mostly the input of the knowledge that provides the adaptivity or pedagogical intelligence along with the more static content. A summary and reference model has been proposed in [12] which also provides general recommendations to designers of such authoring tools. Our approach differs in that jEditOQMath is based on source editing whereas the usage the WYSIWYG (*what you see is what you get*) paradigm is most recommended for micro-level content: the content of LeActiveMath being semantic, its result is multi-faceted and only relying on a browser-like view would neglect the several other perspectives under which to look at the content (in other words *to get it*). In exchange, fast test-run cycles are provided which allow to proof (or *get*) the content under any perspective the learners may be encountering. Relying on the source-and-build paradigm, jEditOQMath does also follow a classical authoring paradigm of mathematical content. Finally, as also recommended by [12], the ontology of mathematical conceptual and content items of OMDoc by their name and XML element names provides a reification that mathematicians can understand well.

Under the view of [2] providing an overview of authoring tools for adaptive hypermedia systems, LeActiveMath is among the multiple indexing systems. Adaptivity in LeActiveMath is mainly within the process of selecting items, within the tutorial component. LeActiveMath is more tight to the mathematical semantic of the macro-structure of items as opposed to the free fragment slicing described in [2]. This can be viewed as a limitation but, at the same time, provides a structure vocabulary that is easy to work with and is more realistic.

3 The authoring environment

The content collection is made of source OMDoc files and static resources. In order to create the OMDoc files the authors are provided with a plain-text editor and they write and read very readable XML sources. The formulæ are input with a more compact format. After input authors can verify resulting views to the content in short edit-and-test cycles.

The current authoring environment of ActiveMath has grown out of the practice of manual editing OMDoc files by developers. The first experience was to edit complete OMDoc files by hand. One realized quickly, though, that OpenMath expressions are much too verbose to be manually input. The approach


jEdit – exponential.oqmath
File Edit Search Markers Folding View Utilities Macros Plugins Help
Error List HyperSearch Results XPath Tool
Templates Tree Structure Browser File System Browser Ant Farm
exponential.oqmath (/Users/paul/activemath/content/LeAM_calculus/oqmath/)
Character Map Console
219,69 4%
(xml,sidekick,UTF-8)- - - - U 19/31Mb

```
<definition id="def_exp" for="transc1/exp">
  <metadata>
    <Title xml:lang="de">Definition der (natürlichen) Exponentialfunktion</Title>
    <Title xml:lang="en">Definition of the (natural) exponential function</Title>
    <extradata>
      <relation type="domain_prerequisite"><ref xref="powerseries_symbols/powerser"/></relation>
      <learningcontext value="university_first_year"/>
      <typicallearningtime value="00:00:30"/>
      <representation value="verbal"/><representation value="symbolic"/>
      <abstractness value="abstract"/>
    </extradata>
  </metadata>
  <CMP xml:lang="de">
    Die (natürliche) Exponentialfunktion $exp$ ist definiert als der Limes folgender
    <textref xref="powerseries_symbols/powerser">Potenzreihe</textref>:
    <p style="cm">$exp(x)  = sum (0._. ∞ ,lambda(n,x^n/n!))
           = 1+ x+ (x^2/2) + (x^3/6) + (x^4/24) + (x^5/120) + (x^6/720)+ dots$
      für jedes $x in bR$ (oder $x in bC$)</p>
  </CMP>
  <CMP xml:lang="en">
    The (natural) exponential function $exp$ is defined to be the limit of the following
    <textref xref="powerseries_symbols/powerser">power series</textref>:
   <p style="cm">$exp(x)  = sum (0._. ∞ ,lambda(n,x^n/n!))
           = 1+ x+ (x^2/2) + (x^3/6) + (x^4/24) + (x^5/120) + (x^6/720)+ dots$
      for any $x in bR$ (or $x in bC$)</p>
  </CMP>
</definition>
```

Fig. 1. A OQMath source being edited, the rendering of which is in Figure 2.

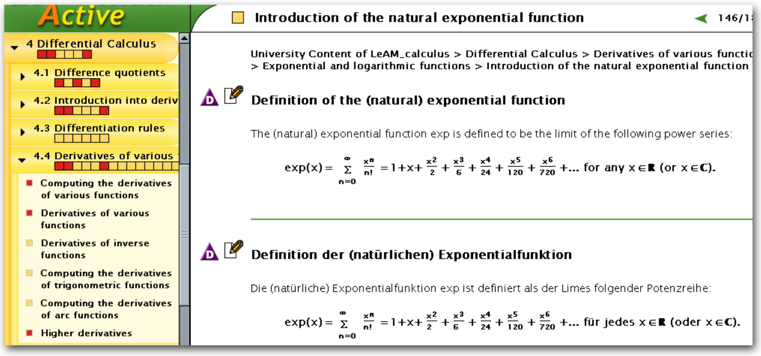
taken by the current environment, called jEditOQMath is thus to process the formulæ using QMath and let the authors continue editing OMDoc files except for the formulæ.

3.1 QMath and OQMath:

Using QMath[4] allows a highly readable syntax to be used for the formulæ. The syntax of QMath can even be more readable than that of TeX since QMath supports the whole range of Unicode characters. An example of input used in the project is depicted in the formula in figure 1. The input of such Unicode characters is non-trivial on most keyboards but is made possible by the definition of *abbreviations* which expand to these characters; because of the common-usage of the TeX language, the abbreviations where taken from it. The mathematical input syntax can be extended by authors by the definition of new notations wrapped in files called *contexts*. Since QMath was written for complete OMDoc documents, a wrapper has been developed around it called OQMath, which extracts the context-declarations and the formulæ and replaces them by their OpenMath translation thus creating complete OMDoc documents.

The OQMath file format is using this processor and the OMDoc DTD. This allows it to be very readable and still be valid XML files. Many editors exist to edit such files, with ample support provided by grammars such as a document type definition (DTD, see [1]).

[4] QMath is an open-source project written in C++ and Bison by Alberto Gonzáles Palomo. More information from `http://www.matracas.org/`.


Active
4 Differential Calculus
4.1 Difference quotients
4.2 Introduction into deriv
4.3 Differentiation rules
4.4 Derivatives of various
Computing the derivatives of various functions
Derivatives of various functions
Derivatives of inverse functions
Computing the derivatives of trigonometric functions
Computing the derivatives of arc functions
Higher derivatives
146/1

Introduction of the natural exponential function

University Content of LeAM_calculus > Differential Calculus > Derivatives of various functic
> Exponential and logarithmic functions > Introduction of the natural exponential function

Definition of the (natural) exponential function

The (natural) exponential function exp is defined to be the limit of the following power series:

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \ldots \text{ for any } x \in \mathbb{R} \text{ (or } x \in \mathbb{C}).$$

Definition der (natürlichen) Exponentialfunktion

Die (natürliche) Exponentialfunktion exp ist definiert als der Limes folgender Potenzreihe:

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \ldots \text{ für jedes } x \in \mathbb{R} \text{ (oder } x \in \mathbb{C}).$$

Fig. 2. LeActiveMath with the example content from Figure 1, presented in English, resp., German (manually added).

3.2 jEdit and jEditOQMath:

The jEdit text editor is such an editor.[5] Its facilities for XML-editing include on-the-fly validation, supported input of child elements and their attributes, and visual folding of sub-trees. jEdit also supports well the Unicode character set, its encodings, display, and input.

As most text-editors, find-and-replace commands are supported, locally and globally. This feature has proven quite important in an ongoing development process where syntax and child elements are changed from time to time.

jEdit was extended to become jEditOQMath by allowing the start-up at the click of a link and the rapid opening and linking of an item through the usage of the drag-and-drop paradigm: the link to an item dropped in jEdit creates, depending on where it is dropped, an OMDoc `ref` element, a link, or opens the source-document of this item.

Content Packaging and Publication In LeActiveMath, the content is organized as a series of content-collections, each with a set of OMDoc files, a set of static resources such as pictures, and a content-descriptor. LeActiveMath uses the descriptor to load the OMDoc file in content-store and identify the books. Once loaded in the content-store, items' content can be accessed individually.

jEditOQMath complements the content-collections with an OQMath directory and *build-file*. The latter is a script to perform the publishing tasks to a running server: the build first applies the OQMath process which outputs OMDoc-files, then sends a reload command to the content-store, finally, invalidates the cache in the author's LeActiveMath.

The OQMath process may complain about parsing errors in the formulæ and the content-store reload process is responsible to resolve all links and check

[5] jEdit is an open-source Java-based text-editor running on contemporary platforms. More information from `http://www.jedit.org/`.

them. Each of these steps may report errors: being part of a build script, error-reporting is done in a similar way as compilers with a quick access to the line of the document where the error is suspected. These errors are added to the XML-validity errors. They are presented with a link allowing an author to go the location to fix the error in one click. This error-reporting activity forms an important management tool which allows permanent control of the consistency of the overall content loaded in the current author's LeActiveMath installation.

3.3 Other Tools Used for LeAM-calculus

The realization of the collection required a few other tools which we briefly sketch here:

Multimedia Elements were created by third-party using classical tools such as Adobe Illustrator for graphics or the NetBeans IDE for graphical design of applets. Their embedding is done in a similar way as the embedding of a picture in an HTML-page.

Because of the text nature of the biggest part of the authored material, a versioning system with text-difference abilities, the classical CVS system, was put to use and has enabled the five persons working on the content to keep in synchronization and to publish their content in demo-servers of the LeActive-Math project. Conflicts sometimes produced by CVS merge operations were not an issue to deal with since the source files were manipulated by authors.

4 Learning to Author

In order to become familiar with the LeActiveMath learning environment as well as to start authoring, a training session was organized. Four authors took part and members of the developers' group held the session.

The first challenges came with the installation requirements for LeActive-Math (Mozilla, a recent Java, Jess, ...) then the configuration of the environment with playable content. The editor was then introduced. The primary experience that authors had with source editing approaches was the TeX typesetting system: this is often done with very simple editors, in comparison to general purpose editors with their numerous functions. For most authors, this session is also where they first wrote an XML-document. The support of jEdit for it turned out to be helpful. Once the usage of the editor, XML editing and the usage of the publication scripts were acquired, the reference mechanism in theories and imports of OMDoc was explained and the error report experimented with.

The input of expressions in QMath syntax that produce OpenMath is then taught: such a lesson starts with the discovery of OpenMath along with examples of the OpenMath content-dictionaries directly from its web-site and continues with an explanation of the QMath notation-*context* files. It ends with an experience of the symbol-reference resolution and rendering of OpenMath expressions in LeActiveMath.

Discovery of the elements possible to input at each point is done, in jEditOQMath, using the XML-grammar described by the DTD, and with the usage of templates which provide pre-filled elements with *blanks* that authors are expected to fill resembling a from.

The training was followed by an introduction into multi-steps exercises [4]. The exercise system was, then, in a very young condition and updates of the grammar were needed to allow correct checking of the syntax. This could easily be done as it only required a few files to be copied.

4.1 Other Authoring Experiences

After the training, the jEditOQMath tutorial was written as an easy introduction for starter authors. Other authoring experiences have been made. In 2004, a mathematics teacher with no experience TeX, or HTML was introduced using the tutorial. After a day, he could do all basic steps. After a week, his input of semantic mathematical formulæ was mature enough to define new symbols. This author has, since then, created large quantities of material in OQMath.

In 2006, the jEditOQMath tutorial was followed by a classroom of computer science students. It took less than 3 hours to get the installations running and their first content be written.

5 The content collection

Since this training, the second author of this article has produced 86 percent of the content for the LeAM-calculus collection. Within the last two years (and approximately 2000 hours) he has authored 185 symbol declarations, 138 definitions, 241 assertions (i.e., theorems, lemmas, propositions, corollaries), 207 proofs, 148 texts (introductions, motivations, notes, etc.), 308 examples, and 268 exercises, where the latter comprise fill-in-blank exercises, multiple-choice-questions, and open exercises, also some search and concept-mapping exercises, as well as multiple-steps exercises consisting of up to 2500 lines of source code. He has enriched the content with more than 23.000 textual links. When converted to a PDF form using LaTeX, the collection is about 450 pages long in German, English, and Spanish. The content covers (nearly) the whole calculus material on differentiation in one variable. As demo material for the adaptivity of LeActiveMath, it is intended for various users in multiple learning contexts, ranging from collective work on pre-recorded books in a grade 11 classroom (about 16 years old) followed by individual homework repetitions up to first year University students who want to rehearse their knowledge in calculus.

Meanwhile we received feedback on the content from university students and lecturers, as well as from high school students and their teachers. In general, this feedback was very positive, most of the users were impressed by the extent of the content. Suggestions for improvement, e.g., demands for additional content, have been incorporated by the authors. In a first field study, approx. 80 grade 11 students have been working with the content in their classrooms and homework

Fig. 3. LeAM-calculus content being used in the evaluations.

repetitions, a picture of which is at Fig 3. According to the online questionnaires they filled in, they were most impressed by the pictures and exercises, as well as by the presentation of the mathematical formulae. On the other hand, they also stated a need for improvement of the exercise hints, which is only natural since of all the scaffolded hints authored for each exercise — ranging from a first, general hint over more detailed tips and partial results up to the complete solution — the software only presented the first one, a bug that is meanwhile fixed.

6 Management Activities of the Collection

As the content was being written it was tested, organized, adapted, retested, discussed, analysed, criticized, searched, ... we consider all these activities as *management* activities since they often require a holistic view at the content collection and how it is used.

Quality Checks One of the first impression authors may have is that the authoring activity has nothing to do with an artistic creation but is much closer to programming. The source editing and build-process both contribute to this impression but the main reason is probably that authors are considered *final recipients of the software* which means that only when their content is integrated can a complete test be done.

The following facets of the content, used in the software, should be tested by authors using both their mathematicians' and pedagogues' eyes:

– The presentation of the textual content interleaved with formulæ, graphics, and other multimedia elements. The correct presentation of formulæ on the web is particularly challenging. The checks must be done for all languages and in the three supported output media and may be stained with platform and browser dependencies.
– The links that are clickable in the presented content, including authored links and symbol references.

– The disposition of books for the planned learning scenarios.
– The correct interpretation of the knowledge about each item encoded in its metadata. Ideally, checking this interpretation should include tests with the learner-model where the knowledge is used to build a domain-map and propagate the beliefs in the learner's mastery or the tutorial component where the knowledge is used to select the items.
– The correct behaviour of interactive exercises including the evaluation of users' input and chosen feedback elements.

This testing activity is typically done in small write-publish-test cycles which are greatly helped by a fast publication mechanism. The fact that this process is enriched with elementary reference and grammar validation is an important safeguard.

In such a research project as LeActiveMath, the authors are among the first users of the software. As a result, quality monitoring is even more important and it has not been rare that authors contribute to the design of the tools by providing early feedback to features.

Exchanging About Content Pieces In order for members of the project to exchange about available content-pieces, care had to be made in the design of the learning environment so that individual items can be addressed by a direct URL. The identifiers of each content fragment can be inserted in such communication forms as e-mails or electronic forums.

The display's possible dependency on browsers and platforms did not prevent exchanges of screen-shots to show the current situation an author was experiencing. The reproduction of such situations was, sometimes a difficult task.

One of the types of interactions where no good solution was found for its description, and where even screenshots have sometimes proved useless, are paths of interactive exercises: one could only describe sequences of inputs which were not includable in e-mails if done in OpenMath using the input-editor.

Since developers are, all, working with mathematical objects in OpenMath, advice to authors was made in OpenMath. Very often, however, authors did communicate with QMath expressions. Such messages took much longer to be interpreted.

Keeping an Overview The content collection is large already but still reflects only a part of the normal calculus curriculum (e.g., topics about integration were completely left out). Tools to help authors and others assessing the content in obtaining an overview of the content are needed:

– The first of these is the automated production of a LeActiveMath book where each page contains the content of each file. This allows quick access to the content being created. The maintenance of other books is done as XML-source editing, relying, among others, on the ability to drag-and-drop a content item within a book's table-of-contents' source.

– Moreover, in order to have an eye on the coverage of the possible target learners and competencies, a catalogue of exercises accessible along their characteristics was done.
– Finally, when writing new notations for the tools of [9], the set of prototype expressions and their associated presentations for all the symbols stored in a LeActiveMath installation are presented. The comparative overview of the symbols, their notations, and argument priorities is presented.

Overall the tools provided to manage content collections of content are blended within the learning environment. This empowers the authors, and probably other expert user, with methods to verify the quality of the content played within the learning environment. As principles of these tools are the fast test-and-edit cycles as well as the one click edition of content under the eyes.

7 Challenges of Authoring Semantic Mathematics

In this section we cover the difficult task of inputting semantic mathematical formulæ. This input is critical to guarantee interoperability with other systems, long term preservation, as well as advanced functions of the learning environment. The tools that we have described above apply to most mathematical content that can easily be input as simple text, which is the case of the majority formulæ in classical mathematics. Let us recall that we expect authors to produce all their formulæ in OpenMath in a conformant fashion.

Authoring Formulæ When first starting to input formulæ, an author needs to find typical patterns of input. This is often found in OpenMath content dictionaries. From these patterns, he can find ways and possibly define input-notations so as to enter such expressions in QMath syntax. Finding these patterns is done by crawling through the content dictionaries and finding the input-notations is done by crawling through QMath context-files. This browsing activity is not comfortable and an integrated search would certainly be helpful. Built-in support for the input of elementary symbols of the formulæ within the editor is also wished. Contemporary integrated development environments provide examples of such support. The process-oriented nature of QMath and OQMath does not offer suitable introspection mechanism for such a support; other parsing facilities have to be provided.

Normal mathematicians conceive their mathematical objects with a good deal of semantics but they are mostly used to write them as presentation. The need to write it semantically is challenging and is made more difficult by the great wealth of mathematical notations.

A typical example that surprises an author during his first authoring experience is the need to use the lambda construct: in the case of the sin function used in a limit, for example: the notation $\lambda.x \sin(x)$ is needed, or, as better understood by mathematicians, $x \mapsto \sin(x)$.
A classical example where semantics encoding is almost at the limit is the usage of ellipsis patterns. Partial results are incorporated in the collection such as the

notation $i = 1, \ldots, k$ but a general formalism and interpretation of ellipses is ongoing research as proved in [14].

Declaring new symbols Fortunately, OpenMath has been designed to be extensible and the declaration of new symbols is possible and even well supported by the OMDoc structure. The declaration of a new symbol goes, as follows:

- First the author defines a `symbol` element, with a name.
- This symbol can then be defined in a single or in several definitions.
- The QMath context files are enriched with notations for this.
- Using them in a QMath expression yields `OMS` elements which are now rendered with default (prefix) notation. One has, then, to define presentation notations.

Refining this presentation is done by the definition of a few `notation` elements associating patterns of OpenMath terms with MathML presentation. These notations are used by the symbol-presentation engine presented in [9]. Being based on OpenMath patterns, they are simple to author and to manage and allow to approach the wealth of notations of the mathematicians. Being based on MathML, they can approach the quality of layout of TeX which is often expected.
Finally, this symbol may be either a mild extension or a completely new concept. If a mild extension, rephrase rules can be authored which can translate expressions using these symbols to expressions in more widespread content-dictionaries-groups.

Why new symbols The introduction of a new symbol can be seen as breaking interoperability since the semantic of this symbol may not be shared by external recipients of them. The introduction of the rephrase-rules allows at least symbols to be introduced for purely presentational reasons or with the intent of a semantic refinement (e.g. to clarify the ambiguity of the symbol `times` of `arith1` CD). An example of such is the declaration, in LeAM-calculus, of the unary-plus sign. This sign actually has no real semantic and was not introduced in the `arith1` content-dictionary. It is, however, important for the presentation as well as for pedagogical reasons. Similarly, $\mathbf{R}_+$ the set of positive real numbers is missing. Experience has proved that many of these symbols are actually translatable in expressions using symbols of the MathML CD-group as defined in [3] which is expected to be supported by many applications..

Publishing new symbols for others to use New symbols may also take a completely new semantic, for example the ray $[PQ[$ has been introduced. Often a publication of this symbol is wished so that others use it. The publication on the web of OMDoc files might be considered sufficient. The OpenMath society has, however, maintained a growing catalogue of content dictionaries on their web-site and this is a recognized central place to discover content dictionaries.

A tool to produce content dictionaries out of OMDoc `symbol` and `example` elements is being realized. This conversion is, however, impossible in full generality. Several features of OMDoc are incompatible with the OpenMath content-dictionary format. This includes the multilingual textual fragments as well as the ability to mix text, links, and formulæ in them. The usage of a renderer of formulæ to text may make the creation of OCD files possible in the future. However, a revision of the OCD format to allow links and multilingual texts might be needed.

8 Other Challenges Encountered

Grammar Specification and Documentation A long requested feature by authors is a complete reference documentation of the set of elements that can be input and their use. Because of the evolving knowledge representation, especially within research projects where software, knowledge representation, and content evolve together, we believe it is needed to have such a documentation bound to the knowledge representation.

The XML DTD of OMDoc can be directly read for this, or can be read with a helper. DTDs are not, however, designed to be enriched with documentation and hence DTD-documentations are very poor.

Using an XML-schema instead of a DTD may be a good solution as this standard has good support and tools for embedded documentation. The migration to such a technology may, however, make the XML-syntax much less readable bringing to the surface, for example, the various namespaces used in OMDoc documents for LeActiveMath. Such an impact may turn out intolerable for the readability of the source files which is a basis of our work.

Input and Testing of Complex Exercises One of the challenges of realizing LeAM-calculus is the input of content for multiple-steps exercises. Multiple-steps exercises form, conceptually a large set of nodes connected by transitions triggered by inputs of the learner. The amount of such nodes, including the diagnostic oriented metadata of each node, has yielded exercises that can be larger than 2000 lines of code where the sole large-scale structure is the interaction graph. Keeping an overview through it is difficult, testing all paths of such is even more difficult especially since, thus far, little indication is provided to a tester as to which condition has been evaluated. For this reason, a graph-based authoring tool is being developed based on the spirit of *authoring by doing.*

Scalability and Distribution of Content The current LeAM-calculus collection is already challenging the content store of LeActiveMath and more content is actually expected. A more elaborate distribution strategy is needed which should allow content collections to be distributed anywhere on the web and be used by the drop of a URL.

9 Conclusion

The realization of the LeAM-calculus collection has proven that a large content collection using tools around jEditOQMath can be written and that a readable XML-syntax has provided the right language level for authors to edit and as basis of the discussion between authors and developers in an evolving software like the one developed in the EU project LeActiveMath.

Other experiences of authoring content for the learning environment using the same tools have proved that even persons with no TeX or HTML capabilities can, within a week or two, be authoring significant content.

The source form of editing may appear primitive to many and, indeed, we received many recommendations to provide a visual authoring tool. The usage of a visual software to edit content would have needed the software to be much more engineered so as to guarantee the trust of authors, an unconditional requirement. Moreover, the evolutionary needs of the project would have required such a visual tool to be permanently adapted to reflect the changing knowledge organization. Moreover, observation of the mathematicians' practices reveals a strong bias, at University level, towards the TeX composition system hence a broad acceptance of the source and build paradigm.

Finally, this paradigm proved very useful as the requirement to track errors raised more and more important. It has been experienced in this project that ignorance of errors reported by the content store yielded easily erroneous behaviour of the latter which triggered buggy error reports of producers or consumers of the content.

The error reporting paradigm proved itself useful and more automated checks are being investigated in order to approach, for example, the completeness in the educational targets' coverage, or the consistency of mathematical macro-structure.

10 Acknowledgements

The realization of this content collection has been done within the EU FP6 project LeActiveMath.[6] The realization of authoring tools have been also partially supported by the BMB+F projects MISS and In2Math.

The authors of the whole set of content of LeActiveMath are Christian Gross, Marianne Moormann, Manolis Mavrikis, and Iain Gibson.

[6] This publication is partly a result of work in the context of the LeActiveMath project, funded under the 6th Framework Program of the European Community – (Contract IST-2003-507826). The author is solely responsible for its content. More information about the LeActiveMath project can be read from `http://www.leactivemath.org/`.

References

1. Bray, T., Paoli, J., Sperberg-McQueen, C.M.: Extensible Markup Language (XML). W3C Recommendation PR-xml-971208, World Wide Web Consortium (1997) Available at `http://www.w3.org/TR/PR-xml.html`.
2. Brusilovsky, P.: Developing adaptive educational hypermedia systems: From design models to authoring tools. In Murray, T., Blessing, S., Ainsworth, S., eds.: Authoring Tools for Advanced Technology Learning Environment, Kluwer Academic Publishers, Dordrecht (2003) See `http://www2.sis.pitt.edu/~peterb/papers/KluwerAuthBook.pdf`.
3. Buswell, S., Caprotti, O., Carlisle, D., Dewar, M., Gaëtano, M., Kohlhase, M.: The OpenMath standard, version 2.0 (2004) Available at `http://www.openmath.org/`.
4. G.Goguadze, Palomo, A., E.Melis: Interactivity of Exercises in ActiveMath. In: In Proceedings of the 13th International Conference on Computers in Education (ICCE 2005), Singapore (2005) 107–113
5. Henry, G.: Connexions: An alternative approach to publishing. In: Proceedings of ECDL 2004 European Conference on Digital Library, University of Bath (2004) See also `http://cnx.org/`.
6. Kohlhase, A.: CPoints mathematical user interface. In: Proceedings of the MathUI Workshop. (2004) Online at `http://www.activemath.org/~paul/MathUI04/`.
7. Kohlhase, M.: OMDoc: Towards an openmath representation of mathematical documents. Seki Report SR-00-02, Fachbereich Informatik, Universität des Saarlandes (2000) See also `http://www.mathweb.org/omdoc`.
8. Kohlhase, M.: Semantic markup for TEX/LATEX. In: Proceedings of the MathUI Workshop. (2004) See `http://www.activemath.org/~paul/MathUI04/`.
9. Manzoor, S., Libbrecht, P., Ullrich, C., Melis, E.: Authoring presentation for openmath. In Kohlhase, M., ed.: Mathematical Knowledge Management:MKM 2005, Bremen, Germany. Volume 3863 of LNCS., Heidelberg, Springer (2006) 33–48
10. Melis, E., Kärger, P., Homik, M.: Interactive Concept Mapping in ActiveMath (iCMap). In Haake, J.M., Lucke, U., Tavangarian, D., eds.: Delfi 2005. Volume 66 of LNI., Rostock, Germany, Gesellschaft für Informatik e.V. (GI) (2005) 247–258
11. Melis, E., Büdenbender, J., Andrès, E., Frischauf, A., Goguadze, G., Libbrecht, P., Pollet, M., Ullrich, C.: Knowledge Representation and Management in ActiveMath. Annals of Mathematics and Artificial Intelligence, Special Issue on Management of Mathematical Knowledge **38**(1-3) (2003) 47–64 Volume is accessible from `http://monet.nag.co.uk/mkm/amai/index.html`.
12. Murray, T.: Principles for pedagogy-oriented knowledge based tutor authoring systems: Lessons learned and a design meta-model. In Murray, T., Blessing, S., Ainsworth, S., eds.: Authoring Tools for Advanced Technology Learning Environment, Kluwer Academic Publishers, Dordrecht (2003)
13. Niss, M.: Mathematical competencies and the learning of mathematics: the danish KOM project. Technical report (2002) See `http://www7.nationalacademies.org/mseb/mathematical_competencies_and_the_learning_of_mathematics.pdf`.
14. Pollet, M., Sorge, V., Kerber, M.: Intuitive and Formal Representations: The Case of Matrices. In: Mathematical Knowledge Management, MKM 2004. Number 3119 in LNAI, (Springer Verlag) Available from `http://www.cs.bham.ac.uk/~vxs`.
15. Ullrich, C.: Tutorial Planning: Adapting Course Generation to Today's Needs. In Grandbastien, M., ed.: Young Researcher Track Proceedings of 12th International Conference on Artificial Intelligence in Education, Amsterdam, The Netherlands (2005) 155–160