

Collection Management in ActiveMath

Paul Libbrecht `paul@(activemath.org|dfki.de)`

Competence Center for E-Learning, DFKI GmbH and University of Saarland,
Saarbrücken, Germany

Abstract. We address the problem of management of content to strive for long-term re-use and quality enhancement for E-learning content. To do so, we propose patterns of content-management that apply to authors (where is stored what and how is it edited?) as well as to the consumers of the content (the perspective under which the E-learning system sees the content in order to offer its services). In the cases of the ActiveMath learning environment, the services that exploit the content are those of a rich web-based personalized learning environment for mathematics: user-modelling, interactive exercises, flexible structuring, and intelligent content selection. All of them are based on the OMDoc documents, enriched in metadata and exercise structure, to support pedagogical activities. The patterns described in this paper include the re-use made possible by projects in shared directories, aggregation, and semantic orientation, the management of references following the approach of imports, and the management of metadata through inheritance. This paper introduces general principles. Each description of a principle is concluded with a description of its implementation in the content storage of ActiveMath.

Introduction

Content for E-Learning is known to be expensive to produce, at the same time it is expected to be of very high quality, both pedagogically and technically. Re-use and management of content appear as two fundamental ingredients to lower the cost by facilitating the ability of an author to simply copy the relevant parts to the relevant places and by making them more handlable. However, for such to happen, best practice to organize the content so as to make it easy for an author to maintain, to find, and to re-use it is still missing.

We propose patterns of content management which facilitate re-use in the perspective of a long-term quality development and in the ActiveMath learning environment in realistic settings. ActiveMath uses the semantic encoding of the OMDoc language as content encoding; this makes the content elements ready to be re-used in many different settings (for example changing the mathematical notation, or exploiting differently the pedagogical functions).

Similar to other learning environments, re-use in ActiveMath is done by aggregation, i.e. by the embedding of content elements into another corpus; many have done so with a duplication approach (e.g. [1]), some, as ActiveMath, perform the aggregation by reference; this is one of the patterns of content-management.

Other patterns fulfill the requirement to track content projects so that communication about the content authoring activity may happen; two others propose methods to synthesize information based on appropriate organisations.

This paper addresses the challenges of manageability of content, proposes patterns to solve them and an implementation to demonstrate their feasibility. Its novelty lies in the recollection of these methods within a set of applied authoring practices working in a deployed learning environment.

Outline

In this paper we describe shortly the central components of the ActiveMath learning environment and how these exploit the OMDoc documents (section 1). This introduces the set of queries addressed to the content to which the storage engine of ActiveMath responds. The other side of the channel is then described, with a description of patterns of content organization which promote re-use as well as managed authored content in section 3. Implementations are described (section 4) followed by a description of alternative approaches (section 5) and open-questions (section 6).

1 ActiveMath as a Consumer of OMDoc Fragments

The ActiveMath learning environment, which is detailed in [2], is a web-based environment for mathematics. ActiveMath achieves a consistent presentation of learning material, together with multiple-steps interactive exercises whose inputs are checked by computer-algebra-systems [3] and the usage of web-based exploration tools for the mathematical knowledge such a function plotter, direct ..., or concept-mapping tool... A user-model is maintained estimating the competencies of the user [4] which is used by the tutorial component of ActiveMath to suggest appropriate paths of learnings among the content items assembled as books. Finally, the whole content is searchable through a search tool for text, formulæ, and metadata.

The content in ActiveMath is encoded in the OMDoc language [5] which allows fine grained content items, somewhat at the level of a definition, an exercise, or a motivation. The content items are made of an identifier, some metadata, and textual content; the metadata provides information about the item such as its type, trained competencies, relationships to other items, or authorship information while the textual content contains text strings interleaved with mathematical formulæ encoded in the OpenMath language [6]; most of the content is made of references to other content items either as hyperlinks or as metadata references (such as the fact that an exercise trains a given concept).

ActiveMath aims at serving simultaneously several classrooms of users, to do so, it needs to present the content rendering in an efficient way; for static content items this is enabled by the presentation architecture which transforms and caches the rendering (in HTML, XHTML+MathML, and TeX).

ActiveMath components read OMDoc fragments by their identifier. Some read the textual parts (grouped in the CMP child elements), some read their metadata (the metadata child element), finally some read the relations from one item or to one item. A short example of content can be seen in figure 1: it is made

```

<exercise id="fib1_pi_Archimedes" for="thm_nested_interval">
  <metadata>
    <Title xml:lang="en">How quickly do Archimedes' intervals converge to  $\pi$ ?</Title>
    <extradata>
      <relation type="domain_prerequisite">
        <ref xref="ex_pi_Archimedes"/>
      </relation>
      <learningcontext value="university_first_year"/>
      <difficulty value="very_easy"/>
      ...
    </extradata>
  </metadata>
  <CMP xml:lang="en">
    Using <textref xref="ex_pi_Archimedes">Archimedes' formulas</textref>
    compute the areas  $\sqrt[n]{A}$  and  $\sqrt[n]{B}$  of the inner, resp., outer
     $2^n$ -gons for  $n$  in (3..7).
  </CMP>

  <interaction id="fib1_pi_Archimedes_problem">
    ...
  </interaction>
</exercise>

```

Fig. 1. An extract of an OQMath source of an exercise (prior to applying the OQMath transformation converting the formulæ between dollar-signs to OpenMath and prior to decontextualization.)

in the OQMath language and is helped by an XML DTD: both of these features enable a relatively readable source format which encodes the full spectrum of the OMDoc language

The exercise system generally reads the entire element, the static presentation mechanism would read the CMP elements to render the exercise invitation within a book page while the element name, the type attribute, and the metadata element are all read to present an enhanced title of the exercise as depicted in figure 1. The tutorial component would query for all exercises for the concept “thm_nested_interval”, which should return this exercise, and for the details of this item’s metadata. The user-model, once a step of that exercise is performed updates its beliefs about the competencies involved in the exercise, this requests the metadata of the step, of the exercise, of the concept, and of some of the items linked to that exercise.

The description above shows that the fragments of OMDoc used in the other components of ActiveMath must be decontextualized, i.e. context-invariant, since they are considered standalone by these components and are often combined with other heterogeneous items of content. This means, for example, that the DTD of the OMDoc document has to be fully *applied* by inserting all its default values within the document served to other components (including xmlns attributes).

```

<exercise xmlns="http://www.mathweb.org/omdoc"
id="mbase://LeAM_calculus/bounded_seq/fib1_pi_Archimedes"
for="mbase://LeAM_calculus/bounded_seq/thm_nested_interval">
<metadata>
<Title xmlns="http://purl.org/DC" xml:lang="en">
How quickly do Archimedes' intervals converge to
<?QMath $$$\pi$$$ ?>
<OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0">
<OMS cd="nums1" name="pi" xref="mbase://openmath-cds/nums1/pi" />
</OMOBJ>?
</Title>
<extradata>
<relation xmlns="http://www.activemath.org/namespaces/am_content" type="domain_prerequisite">
<ref xmlns="http://www.mathweb.org/omdoc" xref="mbase://LeAM_calculus/bounded_seq/
ex_pi_Archimedes" type="include" />
</relation>
<learningcontext xmlns="http://www.activemath.org/namespaces/am_content"
value="university_first_year" />
<difficulty xmlns="http://www.activemath.org/namespaces/am_content" value="very_easy" />
...
</extradata>
</metadata>
<CMP xml:lang="en">
Using
<texref xmlns="http://www.activemath.org/namespaces/am_content" xref="mbase://LeAM_calculus/
bounded_seq/ex_pi_Archimedes">Archimedes' formulas</texref>
compute the areas <?QMath $$$\sqrt{A,n}$$$ ?>
<OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0">
<OMA>
<OMS cd="elementary" name="sequent" xref="mbase://openmath-cds/elementary/sequent" />
<OMV name="A" />
<OMV name="n" />
</OMA>
</OMOBJ> and <?QMath $$$\sqrt{B,n}$$$ ?>
<OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0">
<OMA>
<OMS cd="elementary" name="sequent" xref="mbase://openmath-cds/elementary/sequent" />
<OMV name="B" />
<OMV name="n" />
</OMA>
</OMOBJ>
of the inner, resp., outer
<?QMath $$$2^n$$$ ?>
<OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0">
<OMA>
<OMS cd="arith1" name="power" xref="mbase://openmath-cds/arith1/power" />
<OMI>2</OMI>
<OMV name="n" />
</OMA>
</OMOBJ>-gons for
<?QMath $$$n$ in (3...7)$$$ ?>
<OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0">
<OMA>
<OMS cd="set1" name="in" xref="mbase://openmath-cds/set1/in" />
<OMV name="n" />
</OMA>
<OMS cd="interval1" name="integer_interval" xref="mbase://openmath-cds/interval1/
integer_interval" />
<OMI>3</OMI>
<OMI>7</OMI>
</OMA>
</OMOBJ>.
</CMP>
<interaction id="mbase://LeAM_calculus/bounded_seq/fib1_pi_Archimedes_problem">
</interaction>
</exercise>

```

Fig. 2. The decontextualization and compilation of the exercise fragment in figure 1. Parts which are added are in bold.



How quickly do Archimedes' intervals converge to π ? ★

Using Archimedes' formulas compute the areas A_n and B_n of the inner, resp., outer 2^n -gons for $n \in 3, \dots, 7$.

[Start exercise](#)

Fig. 3. The rendering of the source at figure 1.

This also means that all references need to be absolute so that only string comparison is required to compare identifiers, as we shall see below, this is beyond simple XML storage. The fragment served to the ActiveMath components thus has lost much of its readability for an author while it has gained in portability, a fairly common fact.

All these functions have been grouped within a Java interface called MBaseRef whose methods can be seen at [9]. The approach of a Java interface allows several implementations depending on configurations.

These technicalities reappear in the decontextualization depicted in figure 2 where, for example, the OpenMath standardized semantic has appeared as well as the XML namespaces carrying each their own set of conventions (e.g. those of Dublin Core).

This paper will discuss neither the XML nature of the source of the content, which is mostly inherited from the OMDoc language, neither the practicability of writing XML as an authoring user-interface, for example in comparison to more visual authoring tools. We refer to [8] for a start of such a discussion.

The requirements of the consumers of the MBaseRef interface in terms of performance are dictated by the expected performance of the platform: for about 50 learners, one has estimated the requirement to about 300 MBaseRef queries per second with peeks in such activities as the tutorial components' *creation of a book for your objectives* where the queries for relationships and metadata can be as numerous 10'000 for a single book. To limit this load, several caching strategies have also been devised in the other components ActiveMath, explained in [10] and [11]. See section 4 for details.

Compared to many other content delivery solutions, ActiveMath distinguishes itself by the fine granularity of its content items, an exercise, a definition, ... This is complemented in ActiveMath by the ability recombine the content items into books with a different organization than given in the OMDoc source file. ActiveMath implements this by the notion of book which are written as a *table-of-contents* made of a hierarchical structure of chapters and sections with leaves referencing the content items to be included. This usage is most important for

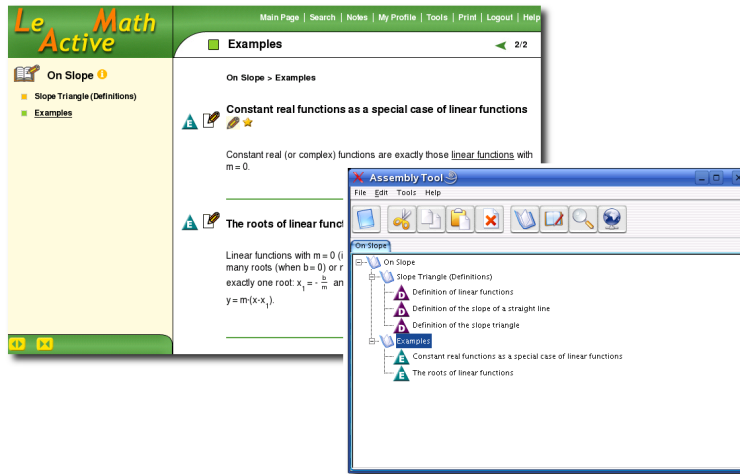


Fig. 4. The assembly tool editing the table-of-contents seen on the back.

the tutorial component’s course generator (whose output is a table-of-contents); also users (authors and learners alike) are invited to re-organize books using the assembly tool: a learner may want this to assemble a portfolio of the important items he has acquired, a teacher may want this to prepare the course material of a given day, an author most probably will use it to create pre-recorded books for the main menu. A view of the assembly tool is in figure 4.

We see that the decontextualization applied in the content-storage is a key ingredient to re-use: it allows users of the content to aggregate content from several different sources within one *book* that is easy to browse. This is one of the elements of a pattern of content management described in section 3.2.

2 ActiveMath Authoring at a Glance

Authoring practices in ActiveMath is described in [8]. It is based on the edition of source files which are part of the content collections; a build process converts these to content-ready files (converting the linear-syntax for mathematical formulæ to OpenMath) which are requested for reload to the content-store. This reload process is indexing the contents, making it ready for the consumption described above. The reload process is where error messages can be raised to authors which provides a first feedback to the content.

The second feedback is obtained by the check phase: after the reload, typically of less than a minute, the content can be seen in action in the learning environment. Provided the author has adjusted his user-profile (the set of information along which ActiveMath adapts its delivery) to resemble his target reader, he can see the exact content as delivered. This is important because the rich set of services offered by ActiveMath should be proved to work sensibly on the content, or for the content to work for the services...

3 Patterns of Management of Content

On the other side of the content storage lies the content edited by authors. In ActiveMath this content is materialized by a set of collection of OMDoc and multimedia files. Each collection is made of a directory containing a content-descriptor which indicates to the content storage where the OMDoc files and multimedia files are to be found, typically, they are in a sub-directory of that directory. In this section we describe patterns that enable long term re-use.

3.1 Re-usable Content Collections as Directories

The first pattern stipulates that content-authors group the contents within projects contained in a directory in a file-system. That directory should contain a descriptor being its entry point but should also contain both the authoring source files as well as the files ready to be deployed to the content-storage of the learning system.

The ActiveMath learning content-storage achieves this by being configured to load a set of collections and offer them to the consuming components. Each collection is a directory with a descriptor and an amount of files of content.

The objective of such a separation is not only to group common topics together but rather to group together a common set of practices and goals. Collections should correspond to the author-side projects, it should have a well defined author (or set of authors) and should be shared with a single license.

Sharing a content collection is then only a matter of exposing that directory in a collaborative place. A directory in a subversion server is a typical place which several author teams have used to share ActiveMath content; sharing a zip archive over the web or email is another possibility.

The interest of packaging both the authoring sources and the content lies in trying maximize the sharing practice so that any sharing recipient has all the possibilities to perform the changes he wishes and to contribute them back to the author if wished. Within an Open Educational Resource sharing model as enabled by open-content-licenses (e.g. Creative Commons licenses) and web sharing (such as the Connexions' authoring-commons analyzed in [12]) the interactions between the sharing recipient and originator may be almost zero at first but become very tight once the exchange of contributions to the content project arise. Such relationships are characterized by purely remote collaborations, often supported by web-forums or email, and being able to speak the same language while talking about the content project constitutive elements is of importance.

This is in sharp contrast to the re-purposing model of Steinmetz et al. [13] where the repurposing is decoupled from the authoring practice: the authoring practice is expected to be that of a professional publishing company whose workflows are not shared with the outside while the recipients of the learning content still intend to re-purpose directly on the delivered content: in the publishing company, a tight coordination can be achieved with a clear separation of roles (for example the roles of redactor, programmers, and designers); in our current experiments however, most authors we have met try to embrace the complete spectrum of activity.

3.2 Aggregation and Project Based Re-use

Aggregation of content from several other pieces to create larger works is another feature proposed in the repurposing framework [13] and that we also propose as part of our re-use framework as explained in [14]. Aggregation is the natural complement to separation in projects, together they allow the scenario of a lecturer re-using last years' content, interactive exercises of a fellow, or polished graphics provided by a professional society, all mixed-in and complemented by the content collection of *this year's course* which does only need to contain the table-of-contents and the complements.

Often the checks described in section 2 lead to modification wishes. This is so common that the right to change contents is often looked at before deciding to re-use. If at all possible (e.g. if the license allows or if the source-edition is doable by the author), the modifications are then done on *appropriated collections*, that is, on content collections which are local copies, for example on subversion checkouts. This is in strong difference to the re-use in the WINDS environment as described in [1] where it is indicated: *Reusability is not measurable in our system as it is based on the copy and paste method that can be applied on various levels – learning unit, material, content block and index.*

The fact that aggregation does not move the content items out of its content collection is crucial to the continuation of the re-use relation: an author may be able to contribute his changes back, if he deems it appropriate, for the sake of a longer quality development of the original content project.

ActiveMath authoring makes this possible thanks to the use of content collections, and of table-of-contents. It supports the easy transition from the check phase in the learning environment until the authoring location in the source by offering a tiny addition to authors in the form of a link that opens in the editor the source of the right item (which the content storage's authoring facilities indicates).

The content format of ActiveMath, the OMDoc language and the underlying OpenMath, also facilitates re-using content because of its semantic nature. The latter enables rendering of the mathematical formulæ taking in account culturally different mathematical ç Thus an inclusion within a book of some re-used content may be done with a change of notations to accomodate the notations of the content item in its *new context*.

3.3 Management of References along the Development Graph

One of the particularly error prone parts of authored content, especially source authoring as described in [8], is the insertion and care of references. In OMDoc, references are extremely numerous:

- the metadata often contains references indicating such relationships as *is-an-exercise-for* or *pedagogically-depends-on*
- hyperlink between content items can be inserted in the text, just as hyperlinks to external web locations
- finally the symbols of any OpenMath expression represent references to the symbol-declaration, constructed from their *cdbase*, *cd*, and *name* attributes.

Because of this large amount of references, a way to keep references readable and verified all the time is necessary. For mathematical formulæ, the usage of the QMath programme allows mathematical notations using a readable linear syntax to insert the references to the symbols.

Hutter and Autexier have solved a similar problem in [15]: they propose to group (formal) mathematical objects within *mathematical theories* and to allow theories to be imported into other theories to extend the second; the authors use this infrastructure to manage the changes of formal content and their impact. The approach of theories and imports has been followed by the OMDoc language and is described in [5]. An example of such a theory extension is realized following the categorical functors of an abelian group to the category of the rings (mapping the group operation to the addition or multiplication of the ring). The namespace-management abilities provided by the organization of content into groups such as theories and into imported groups is what is kept in ActiveMath.

Best-practice in ActiveMath authoring is to enclose all items in **theory** elements and to add **imports** so that the names of items that are referenced are done so with a pointer as short as a pointer to a local item. The figure 5 depicts the typical scenario of an author writing a collection **baseCourse** which includes items from a collection of real-world examples and writes exercises which use symbols of the **units-classical** collection. The names of this example here are kept short for the sake of clarity while the normal usage resembles more the names in figure 1. The references depicted in figure 5 are without imports

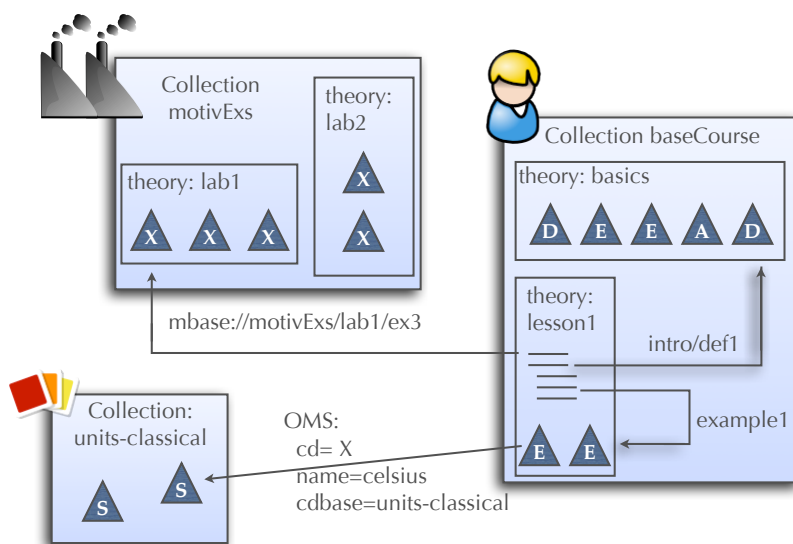


Fig. 5. References between items of various collections: tend to be quite long.

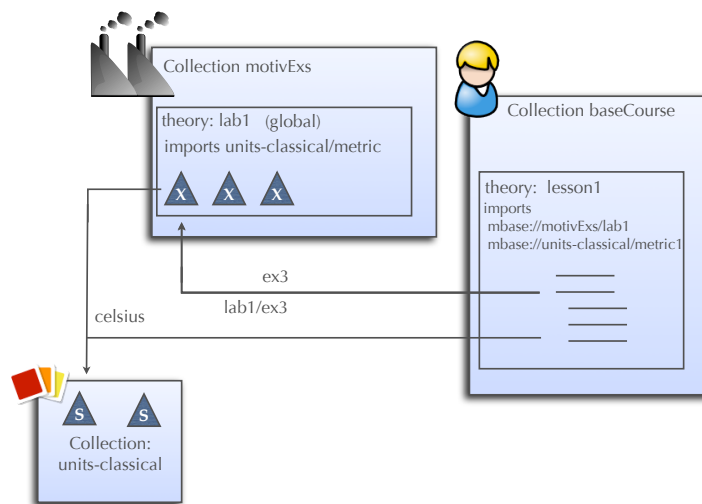


Fig. 6. References with imports between items of various collections are shorter.

and thus can only be absolute (containing collection, theory, and name) while the same references in figure 6 are much shorter (generally only name). Thus we see that the imports shorten the contents of the references thus raising their readability hence management, ease-of-input, and ease of correction.

A simple way to keep references correct is to permanently validate them, as is done in the reload mechanism described above. This validation reports the line-number where this happens which allows an easy “click and fix” approach.

The `imports` elements are part of the context of a content-item: indeed, they define how the references within the items enclosed in the theory should be read. The decontextualization needed for section 1 should thus *resolve the imports* with the following process:

- leave absolute references as-is
- add the collection-identifier of the imported theory of the same name for a reference which contains a theory
- add the collection- and theory-identifiers of the imported theory which contains the name given without theory

We see that this resolution process depends on the contents of the imported theories. To achieve this process, the first implementations were relying on full in-memory representations of all of the complete documents, the current implementation first scans for imports and identifiers and performs the resolution with each file loaded in memory. This move has allowed ActiveMath to store beyond the threshold of approximately 1000 book pages of contents.

Because imports can be recursive (this is the default in the OMDoc language), the set of imported theories for a common set of theories can be kept in one

theory. This theory can be considered as the *set of base materials*. The fact that it is stored in a single place facilitates the adjustments. Among them, the *migration to a branched collection* is a natural adjustment: in the scenario of heavily re-used content-collections, it is highly probable that variants of content-collections are widespread; such a variant would, for example, be the result of a refinement by an external person which could not be brought back into the original work. The variants may be very similar in their identifiers. However, the decision for an author to use another variant may actually come late in the authoring process. If using a single theory as a set of base material, our author only needs to change the name of the collection from one of the variants to the other, and proof the result. An example of such a move may be the appearance of a more modern collection of units symbols (being a recent research development, see [16]): only the import of the `units-classical` collection thus needs to be adjusted to become `units-modern`.

3.4 Metadata Inheritance

We have described above an important authoring artifact for an easier management of references in the content. Another part of content which may take a very large space while not being, visibly, part of the content is the metadata. Metadata is simply the set of data that are not exactly part of the content. Most of the times metadata includes bibliographical elements. OMDoc uses the widespread standard Dublin-Core to this effect.

Typically, however, metadata is constantly the same for many content items in a collection, for example the author of each item of a document may well be the author of the document. Metadata inheritance can be used to this effect: it is used in OMDoc (see [5, section 12.4]), for example, which stipulates that the bibliographical elements missing inside a metadata element should be copied from the enclosing elements' metadata. ActiveMath uses metadata inheritance as specified there and extends it.

The pattern described here is the practice of avoiding the repetition of metadata fields as long as it can be inherited from an enclosing element. This is a pattern that avoids repetition but may also lead to undesired effects if not mastered.

Metadata inheritance can be applied in three different approaches. In all of them, there is a contained item inside a containing item, each have a metadata record and inheritance happens from the containing to the contained item. Inheritance may be of three ways:

- **if-missing**: this is the model of OMDoc 1.2: any given property is inherited only if it no property is provided.
- **merge**: this inheritance merges the properties from enclosing and children. In this way a contributor is an added contributor, an educational context stated in the enclosed element is an added educational context. The merge should not create repetitions of course.
- **nothing**: indicates that no inheritance from enclosing element should happen.

It should be noted that metadata inheritance works on properties and not on element names. Metadata properties are made of all the attributes of the metadata child element: for example the property of a russian translator name may be inherited and not conflict with the property of a french translator name, maybe stipulated at another metadata location.

ActiveMath's metadata inheritance differs from that of OMDoc in several dimensions. First, it allows to inherit the metadata from the collection-descriptor which is a metadata record of the whole collection. Secondly, it offers metadata inheritance by reference to indicate the intent of inheriting metadata from an item at a fully different location. A potential usage may be the sharing of metadata *profiles* which gather all the information pertinent to the typical users in an educational context. It is expressed with a child such as a child of the metadata element `<inherit from="pointer"/>` and should be interpreted as the metadata of the closest enclosing element (ignoring enclosing metadata of enclosing elements of the referenced item).

ActiveMath's metadata inheritance is configurable: all children of the metadata or extradata elements can have the inherit attribute having with one of the above values describing the inheritance ways. This gives flexibility in the inheritance model, allowing to stop the inheritance at parts or even to group items sharing any metadata property within one document, entering only the common metadata on the top of it.

Finally ActiveMath's metadata inheritance applies to the broad spectrum of the metadata of ActiveMath which extends' OMDoc metadata with pedagogical and mathematical properties . For each of them, the inheritance way is as follows:

- Creator, Contributor, Coverage, Date, Description, Format, Keyword, Language, Publisher, Rights, Relation, Source, Subject, Type: inherit-if-missing as specified by OMDoc-1.2.
- Title, Identifier: inherit nothing
- domain_prerequisite, prerequisite, for, counter, references, is-part-of, is_special_case_of: inherit-nothing
- learningcontext, field, semanticdensity: inherit-merge
- exercisetype, exercisepurpose, interactivitytype, interactivitylevel, representation, abstractness, difficulty, misconception: inherit-if-missing
- typicallearningtime: inherit-nothing
- competencies, competencylevel: inherit-merge

Metadata inheritance is, again, part of the context of a content-item, it needs to be applied at decontextualization time so that other consumers see it as part of the content-items' metadata. Because the collection-descriptor and full current document are in RAM while decontextualizing, this operation is performed easily in the content storage indexing phase.

4 Content Storage Implementations

We have described with some details the functions of the content-storage as is an interface between the authors content and the other components of ActiveMath.

In short, it reads the contents in the directories of content collections, doing a first scan of all identifiers and imports followed by a read of each document, each document is *decontextualized*: the XML DTD is incorporated, the identifiers, the relative and imports-powered references are absolutized, metadata inheritance is applied; the resulting fragments, each element that has an identifier, are stored in an index ready to be queried by the other components of ActiveMath: query for full fragment, some attributes, outgoing and incoming references.

The current implementation stores the fragments and the various attributes in a Lucene index. This engine has a proven scalability and the current implementation lives well at it, reaching a mean of 1'000 queries per second. The indexing time may be somewhat long (reaching 20 minutes on a machine with limited disk-speed and limited-RAM for such a collection as LeAM_calculus [8]). However modern computers tend to take about a minute for the same. The important facet of the content storage for authors is the reload facility which, when invoked following the author-side transformations by the build processes, re-parses all changed files, applying the same decontextualization, errors reports, and inheritance.

The content storage of ActiveMath is available within the ActiveMath server which is delivered under the German Free Software License from <http://www.activemath.org/Software>.

The content-storage is also browsable by authors which can, within the web-browser, view the decontextualized content items and browse the contents of theories. Authors may also reload. This tiny front-end has been put to use on <http://commons.activemath.org/> to constitute an authoring commons where all the open-licensed content collections of all authors are shared. The sharing is started by a human request follow by a first install of the collection which is generally shared through a subversion repository. Subsequently authors can request over the front-end to reload which does update. This makes the commons' platform a simple way to publish on the web a previewable form of the content which can follow the evolutions of the content-collection.

5 Alternative Approaches

In general the pattern of a storage interface between the content and the delivery is quite common in most content management systems who use this to optimize their consumption mechanism; the fact that a Lucene index is used it does barely makes it different than an SQL database. One of the storage engine types which, however, has seemed most appropriate has been the XML databases: our experience, however, has never succeeded in having them reach the expected performance trying to offer a too flexible query mechanism.

The decontextualization approach is presented here as a requirement and indeed, it is a common practice: a similar process happens at "compilation time" of TeX-based or wiki-based workflows, for example; it also happens naturally in web-browsers which inject the context of their rendering in order for them, e.g., to resolve URLs from relative to absolute. In both of these cases, extraction of a source to another location may have impact. This decontextualization is the natural translation of a context of authoring which can only exist to a certain

extent. It must stop, for example, in a page combining documents of several sources.

A simple implementation of organization is visible in the default OMDoc software distribution from <http://www.omdoc.org/> which showcases nicely the multiple exploitations of contents: OMDoc files are processed directly by XSL transformations all coordinated by a set of makefiles. Not surprisingly, this method of processing lacks features which ActiveMath requires: imports are almost not used since they imply the XSLTs to read all the full imported theories, and back-references are not queriable. This has resulted in the usage of the simple file-reference with fragment identifiers as being the most used referencing mechanism with imports restricted to particular processing. On the other hand the reference mechanism of ActiveMath, based on a dedicated protocol called `mbase://`, supports imports in its original form: `mbase://collection/theoryB/name` is absolutized to `mbase://collection/theoryA/name` if `theoryA` is imported in `theoryB` and if an item of id `name` only exists in `theoryA`. To our knowledge, attempts are underway in the OMDoc project to enhance the reference schemas,¹ it seems unavoidable to maintain an index of all names and imports for their resolution to be done.

The idea of an intermediate layer that resolves the URIs has also been applied in HELM's getter, which is slightly documented at <http://helm.cs.unibo.it/software/getter/>. The Getter's aim is to serve as gateway to access multiple sources of contents; it delivers the format of the HELM project, CIC-XML, whose internal references may be reformulated so that the getter remains the sole access-points. The getter does not attempt to query links backwards nor does it have a notion of development graph.

Aggregation based re-use is not new. Our approach, aggregation by reference, shares similarity with Ted Nelson's transclusion (e.g. in Transpublishing [18]) although the issues mentioned there have lost actuality because, at least, of the open-content licenses: the latter allow the replication of the re-used content elements so as to ensure its stability and also allow the appropriation of the re-used content which is probably the very first step towards collaborative enhancements.

6 Open Questions

We conclude our paper with the presentation of three ongoing research avenues which we are starting to investigate:

Remote Content Storage A natural wish would follow the getter's practice: a content collection should be remotely accessible. This would considerably ease up the attempts at re-use of a collection allowing an authors' ActiveMath to use (reference and embed) the content of a collection stored in a remote server without indexing it. The remote (XML-RPC-based) access to the MBaseRef API which has been successfully used in client components will be used for that. Using that content storage is certainly slower and less robust but it is a far

¹ Searching communications about the term MMT in the OMDoc spheres may direct the reader properly.

easier installation which can thus be done more often. It should be noted that this is different than Ted Nelson's transclusion because aggregation has not yet happened.

Awareness of the Changes Changes that may happen in the content, by an item change, or by the addition or removal of a content collection, may affect the rest of ActiveMath greatly. The appearance of a new notations may change the way things are rendered, changes in imported theories may trigger an error state, new content items may change the behaviour of the course-generation or search engine... Much of these changes are obvious but their awareness for authors is not cared for much yet. Thus far error conditions are reported as much as possible and events are fired following a content-change so that other components may exploit it; this makes the changed notations be visible almost right-away for example. Method that lead the authors to *see the changes* may be wished. The methods of Plato [17] may form a start but this approach considers very little "multiple views" on content.

Potentially, a related direction may be to actually empower guidance given to authors so that the meaning of theories and imports, as given in [15], is explained in such a way that the semantics of the theory and import elements.

Metadata Reasoning Finally, another avenue is a stronger validation of metadata which can easily become inconsistent while remaining valid along a DTD or an XML schema. A potential avenue is the conversion of the metadata records to individuals of an ontology where axioms may provide a more fine grained and better documented validation.

References

1. Kravcik, M., Specht, M., Oppermann, R.: Evaluation of winds authoring environment. In: Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems, AH 2004, Eindhoven, LNCS 3137. (2004)
2. Melis, E., Gogvadze, G., Homik, M., Libbrecht, P., Ullrich, C., Winterstein, S.: Semantic-Aware Components and Services of ActiveMath. *British Journal of Educational Technology* **37**(3) (2006) 405–423
3. Gogvadze, G., Palomo, A.G., Melis, E.: Interactivity of Exercises in ActiveMath. In: In Proceedings of the 13th International Conference on Computers in Education (ICCE 2005), Singapore (2005) 107–113
4. Faulhaber, A., Melis, E.: An efficient student model based on student performance and metadata. In et al, M.G., ed.: *Frontiers in Artificial Intelligence and Applications*. Volume 178., IOS Press (2008) 276–280
5. Kohlhase, M.: OMDoc – An Open Markup Format for Mathematical Documents. LNCS 4180 Springer Verlag (2006)
6. Buswell, S., Caprotti, O., Carlisle, D., Dewar, M., Gaëtano, M., Kohlhase, M.: The OpenMath standard, version 2.0. Technical report, The OpenMath Society (2004) Available at <http://www.openmath.org/>.
7. Manzoor, S., Libbrecht, P., Ullrich, C., Melis, E.: Authoring presentation for OpenMath. In Kohlhase, M., ed.: *Proceedings of MKM 2005, Bremen, Germany, Revised Selected Papers*. Volume 3863 of LNCS., Springer Verlag (2006) 33–48

8. Libbrecht, P., Groß, C.: Experience report writing LeActiveMath Calculus. In Borwein, J., Farmer, W., eds.: Proceedings of MKM 2006, LNAI 4108, Springer Verlag (2006) 251–265
9. Libbrecht, P., Fuchs, M.: Interface MBaseRef. Java Object Documentation (2003) <http://www.activemath.org/~ilo/redirs/MBaseRef.html>.
10. Kärger, P., Ullrich, C., Melis, E.: Integrating learning object repositories using a mediator architecture. In Nejd, W., Tochtermann, K., eds.: Proceedings of ECTEL'06. Volume 4227., Heraklion, Greece, Springer-Verlag (2006) 185–197
11. Gogvadze, G., Palomo, A.G., Tsigler, I., Winterstein, S.: Presentation Architecture. Deliverable D9, LeActiveMath Consortium (2004)
12. Petrides, L., Nguyen, L., Kargliani, A., Jimes, C.: Open educational resources: Inquiring into author reuse behaviors. In: Proceedings of ECTEL 2008, Maastricht, Markus Specht and Pierre Dillenbourg (eds). Volume 5192 of LNCS., Springer Verlag (2008) 344–353
13. Meyer, M., Hildebrandt, T., Rensing, C., Steinmetz, R.: Requirements and an architecture for a multimedia content re-purposing framework. In Nejd, W., Tochtermann, K., eds.: Proceedings of ECTEL'06. Volume 4227., Heraklion, Greece, Springer-Verlag (2006) 500–505
14. Libbrecht, P.: A model of re-use of e-learning content. In: Proceedings of ECTEL 2008, Maastricht, Markus Specht and Pierre Dillenbourg (eds). Volume 5192 of LNCS., Springer Verlag (2008)
15. Mossakowski, T., Autexier, S., Hutter, D.: Development graphs – proof management for structured specifications. *Journal of Logic and Algebraic Programming* **67**(1-2) (2006) 114–145
16. Stratford, J., Davenport, J.: Units knowledge management. In Autexier, S., Suzuki, M., eds.: *Intelligent Computer Mathematics*. Number 5144/2008 in LNCS, Springer Verlag (2008) 520–535
17. Autexier S. and A. Fiedler and T. Neumann and M. Wagner Supporting User-Defined Notations When Integrating Scientific Text-Editors with Proof Assistance Systems LNCS 4573
18. Nelson, T.: Transpublishing: A simple concept. web-page (1999) See <http://xanadu.com.au/ted/TPUB/TPUBsum.html>.