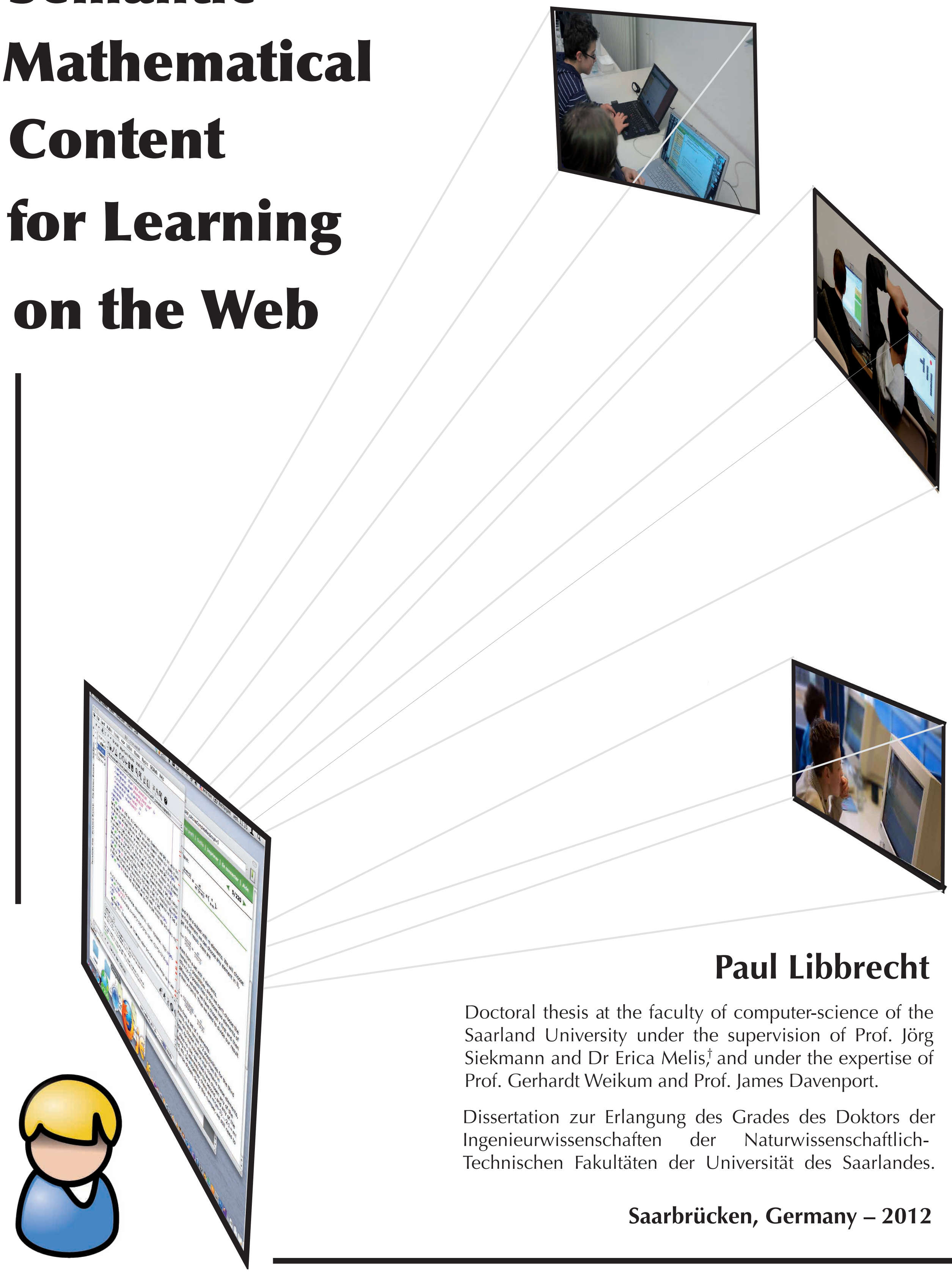
Authoring of Semantic Mathematical Content for Learning on the Web



Paul Libbrecht

Doctoral thesis at the faculty of computer-science of the Saarland University under the supervision of Prof. Jörg Siekmann and Dr Erica Melis,† and under the expertise of Prof. Gerhardt Weikum and Prof. James Davenport.

Dissertation zur Erlangung des Grades des Doktors der Ingenieurwissenschaften der Naturwissenschaftlich-Technischen Fakultäten der Universität des Saarlandes.

Saarbrücken, Germany – 2012

This doctoral thesis has been defended on July 30^{th} 2012 at the University of Saarland under the supervision of a committee appointed by the faculty dean, Prof. Mark Groves: Prof. Jörg Siekmann (reporting chairman), Prof. James Davenport (reporter), Prof. Gerhard Weikum (reporter), Prof. Anselm Lambert (chairman), and Dr. George Goguadze (faculty member).

Colophon

This doctoral thesis is typeset using the `pdflatex` programme, a contemporary part of classical TeX distributions (see `http://tug.org/`). This programme is run through TeXshop, an application for TeX on MacOSX (see `http://pages.uoregon.edu/koch/texshop/`).

The source of the thesis has been partially written in TeX sources (chapters 2, 5, and 9, as well as the chapters without numbers) and partially using OmniGroup's OmniOutliner, an outliner for MacOSX (see `http://www.omnigroup.com/omnioutliner`), followed by a hand-made XSLT stylesheet to convert an almost visual source to its TeX counterpart.

This thesis is typeset using the font `urw-classico`, a clone of the font Optima realized by Bob Tennent for the TeX users; see `http://www.ctan.org/tex-archive/fonts/urw/classico/`. The main body of the text is typeset in 10 pt size.

About the Cover-Page Graphics

This picture, made of a collage of photographs by myself, Carsten Ullrich, and Edgar Kessler, expresses the projection relationship between the author at work with his authoring tools and the various learning situations.

Ownership and License

The text in this thesis has been entirely written by me. At times, I cite research of other persons using the normal citation mechanism: in this thesis, I am using a short key made of the initial part of the author names in square brackets followed by the year, as in [ML71]. The key refers to the bibliography appendix where all the details of the publication are given and, as much as possible, a URL to access the publication.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbst- ständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Abstract

This thesis explores how a mathematician can create content which lets students learn via ActiveMath. This software runs on the web, and employs semantics mathematical technologies and artificial intelligence to support the student. The thesis describes not only a tool to edit content but also studies the workflows of authors when creating content.

An important authoring paradigm introduced here is called WYCIWYG – *What You Check is What You Get*. This places the target learning environment as the central concern. Indeed, the learning environment, which is the assembly of content and software, represents the final product of the authoring work. This thesis exploits and develops classical paradigms: copy-and-paste, search, and collaborative authoring.

To support the authoring workflows, several software pieces have been implemented and tested: within an editing tool and the ActiveMath software. The following work describes their usages, their technical facets, and case studies of how current authors have used them in their day-to-day activities.

Zusammenfassung

In dieser Dissertation untersuchen wir Möglichkeiten, die einem Mathematiker zur Verfügung stehen, um Lerninhalte für ActiveMath zu gestalten. Diese intelligent-adaptive Lernumgebung nutzt semantisch-mathematische Technologien und Methoden der künstliche Intelligenz, um den Lernenden zu unterstützen. Diese Dissertation beschreibt nicht nur ein Werkzeug zur Bearbeitung des Inhaltes, sondern auch typische Abläufe und Prozesse bei der Inhaltsgestaltung.

Ein wichtiges Paradigma für die Autorierung nennt sich WYCIWYG (für *What You Check is What You Get*: *Was Sie prüfen, ist [das,] was Sie erhalten*). Es rückt die Lernumgebung in den Mittelpunkt, denn letztlich ist diese als Zusammenspiel der Software und der erstellten Inhalte das Endprodukt, mit dem der Lernende arbeitet. Kopieren und Einfügen, Suche sowie kollaboratives Autorieren sind andere klassische Paradigmen, die in dieser Arbeit aufgegriffen und weiterentwickelt werden.

Zur Unterstützung des Autorierungsprozesses wurden mehrere Programme, sowohl in der Lernumgebung ActiveMath als auch in einem Texteditor, entwickelt und getestet. Diese Doktorarbeit beschreibt deren Gebrauch, technische Aspekte, sowie Fallstudien von Autoren, die sie in ihrer täglichen Arbeit verwendet haben.

Résumé

Dans cette thèse, nous explorons les différentes voies qu'a un mathématicien pour créer un contenu permettant d'apprendre avec ActiveMath: apprendre les mathématiques avec un logiciel du web qui emploie les technologies des mathématiques sémantiques et de l'intelligence artificielle pour assister l'étudiant.

Un des paradigmes important de la création de contenu s'appelle WYCIWYG (*What You Check is What You Get*: *ce que vous révisez est ce que vous obtenez*), il met l'environnement d'apprentissage au centre. En effet, cet environnement, assemblage du contenu du système, est l'objectif final de la création. D'autres paradigmes classiques sont exploités: le copier-coller, la fonction de recherche et la création collaborative.

Pour contribuer au processus de création, plusieurs logiciels ont été réalisés et testés, tant dans un outil d'édition que dans ActiveMath. Dans cette thèse, nous décrivons leur usage et leurs aspects techniques. En outre, un chapitre est consacré aux études de cas de plusieurs auteurs qui les ont utilisés dans leur quotidien professionel.

Acknowledgements

I would like to thank my thesis advisers, Erica Melis[†] and Jörg Siekmann, who always tried to push the advancement of this research and have given it a frame towards its reality. I am particularly thankful to Erica Melis for her permanent opening to my questioning and inclination to debate, and the professional replies she always made to them; I am particularly thankful to Jörg Siekmann for his impressive enthusiasm about the subject.

The team where this research has happened was intrumental to this thesis: the many authors that patiently tried and accepted or rejected authoring methods, as we report in the chapter 9 as well as the team at the ActiveMath group. I would like to particularly thank Eric Andrès and Michael Dietrich, for having taken over the duty of teaching authoring. Thanks to Carsten Ullrich, Stefan Winterstein, George Goguadze, Martin Homik, Michael Kohlhase, and Vladimir Breznev for active and fruitful discussions around ActiveMath. Thanks to all our students for their curious and brave implementation attempts, in particular Shahid Manzoor and Yecheng Gu. The detailed list of software contributions is described in the appendix A.

I'm indepted to my committee for their reviews. I would like to particularly thank James Davenport for his proactive suggestions which have uncovered the depth of the challenges several times.

I would like to thank my parents and family at large for their impulse into this intellectual entreprise. They have contributed to create the curiosity and tenacity needed to attack and discover such research.

This thesis is largely due to the patience of my family which saw me spend long nights and week-ends on the topic while the weeks were busy with more business-oriented works (such as research projects). My wife Corinne, in particular, has supported each advance by encouragements and acclamations. They have even brought my kids to pique me on a regular basis to ensure that the thesis was the focus of work! Thanks for their patience while it has taken much longer than one would have wished or planned.

[†]Erica Melis has directed this thesis until 2008. She passed away in 2011.

Contents

Introduction

Wege entstehen dadurch,
dass man sie geht.
(Paths are created by walking them.)
(citation attributed to Franz Kafka)

The objective of this thesis is to investigate tools that allow mathematicians to write content for the ActiveMath learning environment.

The research investigations described in this thesis represent about 10 years of partial-time investigations devoted to the authoring toolset and coaching in parallel to main projects focussed on the ActiveMath learning environment itself. The authoring and coaching investigations have been conducted throughout the years with a diversity of tools and versions of the learning environment, as well as training and supporting users of the tools in person and remotely.

What is authoring? Despite its simplicity, this question does not have a fully defined answer, but we approach its multiple facets in Chapter 1. Authoring involves using tools to create ways of using the computer-based enviroment that results in learning for the user. The tools and their patterns of usage are what we are trying to describe in this thesis.

Where it happens

The target beneficiaries of this research are the authors who are expected to be practicing mathematicians with an interest in the use of technology for learning. We have met the following characteristics of authors, which are worth highlighting: mathematicians generally use blackboards as a major thinking and explanation method even if actively using computers, which reflects their traditions of writing. Another strong tradition that I have observed is that of the domination of the TeX typesetting system – a tool that has satisfied generations of mathematicians by its impressive typesetting quality. Any new tool is often compared to TeX, which can present quite a challenge. Finally, all the authors I have met are persons that wish to master their computing environment, in that they want to be able to realize each step of their authoring work and know its results in a tangible and trustable fashion.

These characteristics are fuzzy and rather heuristic but they summarize well the challenges that occur when researching authoring tools that are to be adapted to their intended audience instead of attempting to impose artificial workflows.

During work on this thesis, I have been researching and developing for the authoring tools and the learning environment, as well as training and supporting users of the tools in person and remotely. I have also been advising occasionally on the development of the content.

Conventions

In this thesis, I use the singular first person (e.g. *I use*) when it is clear that I, the author, am the only speaker. However, I also use the plural first person, when the reader and author are seen together (e.g. *we shall see*).

An old tradition in human computer interaction says that one should use the feminine third person to speak about the user (one says *she, the user*); this tradition has been kept in many of the papers presenting the ActiveMath learning environment, as it avoids the need to speak the two user genders. In this thesis, I have chosen to extend this convention. I shall use:

- *she, the learner*
- *he, the author*

As can be seen in the case-studies (Chapter 9), where the majority of authors are female, this does not mean any gender bias.

Organization of this thesis

This thesis is organized in three main parts, each with individual chapters that focus on specific aspects:

- The first part, called **Context**, sets the stage with a detailed summary of the state-of-the-art around authoring and ActiveMath. It should help the reader that is not familiar with authoring or with ActiveMath understand the challenges.

- The second part, called **Authoring for ActiveMath**, describes in detail the ideas that constitute this thesis. Also, following a tradition in the ActiveMath group and of many other computer science research teams: all research published about is to be validated by a functioning implementation; thus, reports on the implemented techniques are also presented.
- The third part, called **Ease of Use**, is a transition from a report on implemented enhancements to a report on the evaluation of the features by learners and authors.

In each chapter, we follow the classic pattern of presenting the pre-existing research, our contribution, and then future research.

These three parts are enriched by several appendices which can support the readers to clarify their understanding. Each computer science term that is used in many places is shortly explained in the **glossary** using a star suffix e.g. HTML*. A bigger picture of the software including who has created what and an architectural view can be obtained in the appendix A about **software contributions**; the wording of the software as it appears to authors is listed in appendix appendix B, about **software functions**. The main concepts of this thesis are listed in the **index** so that they can be found easily. A **bibliography** provides links to cited literature.

Guide for the rushed readers

Each chapter can be read in isolation; only a few referrals to other chapters would be needed for support. Diving into the thesis can begin from the first chapter onwards or by using more direct routes:

- by concept, via the index
- by the table of contents, to find a desired chapter
- by the software functions appendix which links each command to the relevant section of the thesis

Users of the electronic versions of this thesis should notice that the navigation between the different parts of the thesis is made easier with hyperlinking: all glossary terms, commands, and bibliographic references are clickable hyperlinks, similarly for the table of contents, index, as well as the number in each chapter and section reference.

Publications related to this Thesis

Several chapters of this thesis have seen the light first as a timely academic publication before being matured, updated, and integrated into this thesis. They are mentioned at the end of the introduction of each chapter. For completeness, the list is the following:

- **Chapter 3: WYCIWYG: Authoring with jEditOQMath**:
  [Lib10] Paul Libbrecht. What You Check is What You Get: Authoring with jEditOQMath. In *Proceedings 10th IEEE International Conference on Advanced Learning Technologies.*, pages 682–686. IEEE, july 2010.
  Contribution: sole author.

- **Chapter 5: Getting Access to Mathematical Content**:
  [LM06]Paul Libbrecht and Erica Melis. Methods for Access and Retrieval of Mathematical Content in ActiveMath. In Nobuki Takayama, Andres Iglesias, and Jaime Gutierrez, editors, *Proceedings of ICMS-2006*, number 4151 in LNCS. Springer Verlag GmbH, september 2006.
  Contribution: main author.

- **Chapter 6: Publication and Re-use of Content for Web-Platforms**:
  [Lib08] Paul Libbrecht. 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, Sept 2008.
  Contribution: sole author.

- **Chapter 7: Helping to Input Formulæ through Transfers**:
  [LJ06] Paul Libbrecht and Dominik Jednoralski. Drag and Drop of Formulae from a Browser. In *Proceedings of MathUI'06*, August 2006. Available from `http://www.activemath.org/~paul/MathUI06/`.
  Contribution: main author.

  as well as: [LAG09] Paul Libbrecht, Eric Andrès, and Yecheng Gu. Smart Pasting for ActiveMath Authoring. In *Proceedings of MathUI'09*, July 2009. Available from `http://www.activemath.org/workshops/MathUI/09/`.
  Contribution: main author.

- **Chapter 9: Case Studies**
  [LG06] Paul Libbrecht and Christian Groß. Experience report writing LeActiveMath Calculus. In Jon Borwein and William Farmer, editors, *Proceedings of Mathematical Knowledge Management 2006*, number 4108 in LNAI, pages 251–265. Springer Verlag, aug 2006.
  Contribution: joint author.

Part I

Context

Chapter 1

Authoring an E-Learning System

1.1 Introduction

This thesis focusses on the creation process of a *learning experience* using the ActiveMath learning environment. In order to delineate the role of the creator, as opposed to the role of developer, supporter, or even advanced user, we start this thesis with an attempt to define the authoring work: that of a creator of online educational content for mathematical education.

We wish to define the work of authors because the achievement of an online educational experience is the outcome of the work of many roles:

- The **software architects** have designed the learning environment with a broad idea of its various potential applications.
- The **software makers** have implemented it in a way that is slightly more specific and delivers concrete actionable software. Neither of these categories indicate how and where the software would be used, nor what it would present. But they have typical examples in mind.
- The **presentation-engineers** implement the *views* in a slightly more specific fashion: they parametrize the rendering of the content so as to present it appropriately. They make use of existing materials for comparison.
- Just before the end, the **creator** may be writing content which he sees as appropriate for a planned learning experience. He knows partially how the content will be used and, typically, has deployed it himself in his teaching work.

- At the end of this line is the **editor**: This role, which is mostly combined with the authoring role, is responsible for "all the rest": i.e. to prepare the environment so that the educational content and information is correctly presented and the computer-based learning activity is fully functional and accessible for the target audience (the learners) in all the aspects." It is generally assumed by **teachers**, and/or **tutors** and can be supported by system **administrators**.
- Finally, all this is directed to **learners** who *consume* the contents in a way they find appropriate.

We focus on the creation work of content for **web-based** environments only. This restriction underlines the role of the editor as that of a server administrator and gives the learners the possibility to use the content any-time and any-where.

1.1.1 Existing Definitions of Authoring

Not much literature has been published on modelling or delineating the authoring work without being affiliated with specific software. The closest we find is KBT-MM, a rich model describing the organization of data for *knowledge based systems* proposed by Tom Murray in [Mur03]. The approach of this chapter is more concrete than that of KBT-MM, as it focusses on user-interfaces and interactions with the content. In comparison, the model of [Mur03] is a knowledge-representation model: The research in this thesis focusses little on knowledge-representation, which we mostly inherited through prior art and consensus work described in Section 2.2.

Similar *authoring knowledge models* include other chapters in [MBA03], the LAOS layers [CdM03] of Alexandra Cristea, the FOSP method [Kra04] of Milos Kravcik, and the knowledge lifecycle of Millard and Davis [MTD$^+$06]. All of them are more models of content organization rather than descriptions of the work of the authors. Moreover, they all have strong implications on the learning environment (e.g. indicating what is formulated where).

Among the more recently published books on authoring tools for learning is the book of Loos et al. [LZC08] which describes an approach to organizing the content creation as a business process, typically involving multiple specialists. This book starts with a concept defining chapter [Zim08] which takes time to define the objects of the learning content but not the authoring tools or work. Moreover, this book and chapter focusses on the organization of a multi-person creation process with defined specialties, an organization that we have rarely seen in the authoring of mathematical content.

The English WikiPedia entry about authoring tools, at `http://en.wikipedia.org/wiki/Authoring_tool`, is very insubstantial as it seems to merely list a selection of software which could all, in various ways, be considered e-learning authoring tools. There seems to be no reasons to apply this restriction in the selection.

Among the recognized references in the foundations of E-Learning is the book by R. Schulmeister [Sch07], which allocates a section to *authoring systems* - however the section (almost all of Chapter 4) is a historical survey on what authoring systems were envisioned as during a time when Skinnerism prevailed. They have their origins in authoring methodologies for *programmed instruction*. This chapter links to multiple studies that form predictions on the contribution of authoring systems to the success of technology enhanced learning; yet none converge, and almost none show that one particular authoring system has been widely and successfully used. The same chapter of this book, however, highlights the importance of empowering the didactic imagination to create successful learning experiences.

A web-search for authoring tools delivers a large amount of definitions which generally revolve around the same aspects: a tool to create content that is ready to be delivered to others. As a typical example, the *Business Dictionary* indicates: *Software that allows (usually non-programmer) users to create their own courseware, web page, or multimedia applications and the associated navigating tools*. (see `http://www.businessdictionary.com/definition/authoring-tool.html`). We shall rely on this definition, and hence formulate:

The authoring work for learning is that of persons that create the content ready to be used in learning activities.

In this chapter we inspect a diversity of authoring tools used in learning so as to expand this definition into a model of the authoring interactions.

1.1.2 Authoring in other languages

Looking at how authoring as a concept is expressed in other languages can help us understand neighbouring views of this work.

In French, authoring tools are commonly called *outil de création*. This underlines the creative nature of the work; the translation *outil d'auteur* is also used but rarely.

In German, the verb *to author* is often translated as *entwickeln*, which rather means *to develop*; it has the same neighbour meaning as *to develop* in English: to unfold. This verb is also sometimes translated as *gestalten*, with the closest translation in English being *to set-up*; this translation depicts the activity as being *elaborative* e.g. as one would set-up an exhibit. The closely related name *Gestalt* originally describes the physical posture of a person.

1.1.3 Outline

In this chapter we first describe what we believe an author expects of an authoring work, what resource he has to achieve it and what the authoring tools should offer him. A few extreme cases follow: WYSIWYG, wizards, and knowledge encoders.

We then review seven authoring tools and describe them along the *spaces* where the content elements are reachable by the authors and the possible interactions an author has with them. These spaces are the interfaces of an author to the content. We preferred the word space because it describes a place where an author can immerse themselves and manipulate things, return, and then immerse themselves in another space of their choosing. The spaces provide a model of the authoring work which the other chapters have attempted to follow.

1.2 Expectations of the Authoring Work

As we have mentioned above, we consider the authoring work as that of the two parts of usage of the software before the learner's usage: the usage of software by a person wishing to set-up a learning experience. The authoring work begins with a project of educational knowledge that will be encoded, assembled, and laid out for consumption by identified target users. The realization of this project is done through the usage of the authoring tools.

The main persons involved in the authoring work will be called **authors**, either in the role of modifiers or creators. They are the users of the authoring tools that manipulate the learning environments in such a way that it is possible for learners to interact with the resulting environment in a manner that allows them (the learners) to receive the authors' intended experience.

Compared to the practices I could have observed in multimedia and print design agencies, the authoring activity for presenting educational content for mathematics is considerably more **solitary**: the author is a person that wishes to be able to express everything they need to via the use of the tools; any inability to reach a desired state that, for example, requires one to wait for a response from another person, will be considered to slow down the process and will be avoided. This requirement has two very strong implications: there will be many previews and and many attempts.

The authoring tools must be able to give the author an idea of the expected results; typically this is done by the help of **create and preview cycles**. Also important is that the previews need to provide a *high fidelity*: to occur in a way as close as possible to how an expected learner will consume it. To quote REDEEM's presentation [AMG$^+$03]: *We agree with Sparks et al (in this volume) that authoring tools will be more useful if they easily support progressive authoring.*

Another implication is that the authoring work is **more exploratory than methodic**: the authoring tools are all the authors have as ways to realize their encoding, they will be bent and twisted in all possible ways to achieve the desired results. The exploratory nature is opposed to a deep science of authoring: the authors expect the work to be simple and will learn as they progress to new achievements.

A metaphor we would dare to use to characterize a successful authoring tool is that of a tool that can be run on an island: the author is alone and without connection to the world and wants to achieve most of the creation and still feel sure that the created content will be easily deployable to the expected consumers.

1.2.1 Extreme Realizations of the Expectations

In this section we present approaches that are somewhat extreme interpretations of the expectations described above. They represent incomplete solutions which are, however, widespread for particular authoring tasks.

One of the extremes pushes the required short preview cycles to the lowest extreme: the **WYSIWYG** paradigm, *What You See Is What You Get*. This paradigm is implemented in many word-processors and desktop design programmes (such as MicroSoft Word, Adobe Illustrator, or QuarkXPress) and it stipulates that the editing interface includes a faithful view of what shall be obtained. This is short-sighted for web content, and particularly adaptive ones, where all the possible things one can *get* cannot be packed within a view: for example, simple HTML* authoring needs checking within several browsers and also requires the verification of links; authoring for RSS feeds needs checking (or at least consideration) for consumption in many different devices of varying screen-width...

Given that simplicity is a very important facet of making the software accessible, a simple method to elicit the required information from an author is to *ask them questions* within a sequence of modal dialogues. This approach has been often deployed in the form of **wizards** whose *magic wisdom* allows to create the content with a single touch. In practice, wizards certainly offer a very accessible start to work, but are generally frustrating to users that wish to fine tune: the amount of texts to be read gets quickly unwelcome and the modal dialog, step-by-step approach blocks any other manipulation before the wizard is concluded, discouraging quick editing cycles.

An extreme measure, that is far from easy but justified by the expectation that *one could do it all* is **programming**. It is often tempting for authors to go under the hood and to wish to change the behaviour of the programmes. Even though such modifications are welcome in many situations, at this point they are no longer authoring: they are taking part in the software lifecycle, between development and configuration, with a much tighter integration to the development team and a greater difficulty to share or mix the result of these modifications with others. Some softwares allow content to be bundled with programmes but this is a problem for the contents to be shared safely in a world of distant discovery such as the web.

Finally, a type of contribution which has been often considered part of authoring is that of **knowledge encoding**; for example inputing the rules that dictate how a user-model is updated as in [BSS$^+$07], the pedagogical strategies involved in content selection (as developed in [Ull07]), the rules of an expert system such as the Jess-rules of the cognitive tutors in CTAT (see [AMSK06]), or exercise feedback strategies as in [Gog11]. These tasks can be considered part of authoring if simple enough and if focussed with the creation of content. The extreme however goes in the direction of making this an abstract task independent of the content: when this knowledge input becomes an abstract task the lack of direct applications requires knowledge engineering approaches; for example, it needs clear requirements and identified test scenarios.

1.3 Sample Authoring Tools

We have selected seven authoring tools, some classic, some new, that we shall use for comparison:

- **MediaWiki**, the software behind the Wikipedia encyclopedia, is a web-based wiki: it can be considered an authoring tool for editing pages of hypertexts using a simple text syntax. The authoring tool is bundled inside the delivery architecture. See `http://www.mediawiki.org/`. MediaWiki is used in a large amount of educational projects.

- **TeX** and **LaTeX** are two processors that transform plain text files into visual documents with unsurpassed qualities for mathematical formulæ. The syntax is easy to learn and sophisticated layouts can be built with them. Their main usage is for paper-oriented documents with the static graphical aspect coming foremost. See `http://www.tug.org/`. TeX is widely used in learning activities and comes often as first choice for university-level mathematics lecturers to produce assignment sheets.

- **Adobe Illustrator** is one of the most classical graphic design software used by thousands of users around the world. It features a WYSIWYG* approach centred around the delivery of a static image although it supports a few extras in the direction of interactivity. See `http://www.adobe.com/products/illustrator/`. Illustrator is widely used among graphics designers and is often used as a picturing tool for mathematical illustrations.

- **CTAT** example tracing (Cognitive Tutors Authoring Tools) is an extension of the NetBeans and Flash* Development Environments together with runners that allow authors to create interactive exercises with feedback to multiple learner-inputs. The authors edit the views using the environments graphical component systems and manage states on a *behaviour graph*. See [AMSK06] or `http://ctat.pact.cs.cmu.edu/`. Usage of CTAT is growing, including its use for school mathematics, the publications mentioned show several usages of it.

- **Interactive geometry systems** such as Cabri (see `http://cabrilog.com/`) or Cinderella (`http://cinderella.de/` or [Kor99]) are tools that allow the creation of dynamic geometry constructions which are shown or experimented with on a computer. The object of the authoring is a geometric construction whose free handles can still be moved to explore the geometric relationships. Interactive geometry systems such as these are widely used by the school practicing teachers and are among the interactive experiences that are easiest to author.
- the **AHA!** adaptive environment which, together with its authoring tools, constitutes a shell to create adaptive web-applications with a fine-grained control on the user-model. See [BSS^{+}07] and `http://aha.win.tue.nl/`. AHA! is recognized as one of the model adaptive systems.
- **jEditOQMath** is my authoring tool for the ActiveMath learning environment and the central user-interface of this thesis. It is described in Chapter 3. jEditOQMath is based on the edition of source texts that represent the semantic documents in OMDoc and is tightly coupled with ActiveMath so as to make the authoring result visible.

There are a considerably larger number of authoring tools available than the ones detailed above, but these seven samples present a wide spectrum of the typical aspects of authoring tools. In the remainder of this chapter, this wide range will provide us with thorough coverage as we explore in detail the various interfaces where an author meets content.

1.4 How is Content Accessed

Let us call ***content space*** any user-interface where content can be considered to **be situated**, that is, where any user would *find* it. This includes content representations of different encoding natures and with different functions (browse, read-about, write, arrange, update,...). The word *space* has been chosen instead of user-interface so as to highlight the fact that an author enters that space and manipulates it. The word instrument could also have been chosen following [BL00] but using this metaphor would have hidden the situational aspect.

1.4.1 Non-Computerized Places

Content and its related human-knowledge is present in many places other than stored in a computerized medium. For example the mind of an author or learner, a scrap paper, a blackboard, or a textbook. Similarly, (ideas for) content-fragments are often sketched in places external to an authoring tool (an outliner, a topic-map, ...).

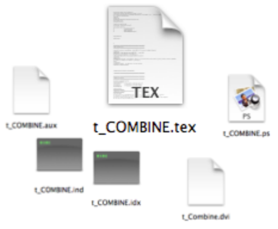
The authoring work could be considered a transfer from these media until an organized storage form which the learning environment can use.

This transfer relies on many different knowledge sources which are often known to the e-learning author. This includes a satisfactory knowledge of the domain to be studied, of the content to be encoded, of how learners will receive it, and of the methods to express and present it. It also relies on the pedagogical theories that support the author in the ways the domain will be taught and is particularly important for pedagogical practices that are known to be applicable to the learning environment being authored for. Finally, it also relies on the experiences the author has had with the learning environment; these experiences are the bases of the expectations of the author.

1.4.2 Authoring Sources Management

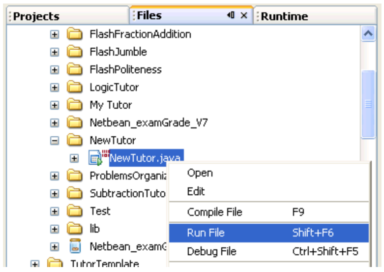
At the very start of the computerized authoring work stands the manipulations of the container objects that represent the entities of authoring.

In most cases, these are files stored in directories. Very often these files are well known to the authors who can recognize them by name. The mapping of an object to a file is not always one-to-one.



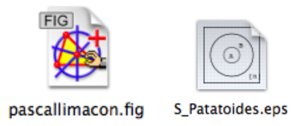
In web-based authoring, the container of authoring objects are often less tangible. MediaWiki authoring is based on the reading work: one edits the page being read, or a section of it.

TeX often works within one directory where related files are also included or close-by. In particular `.aux` and `.log`, and `.bbl` files are all by-products of the TeX runs. The figure on the right shows an example of how one would find these in the MacOSX finder; these files were found in the editing process I took part in for the course-book of Combinatorics at UQÀM.

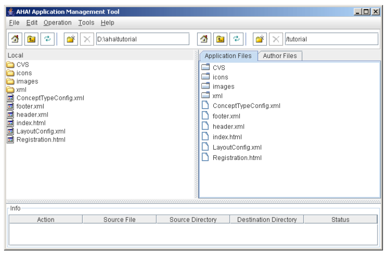


CTAT is based on *development environments*, those environments that are used by developers to create software. CTAT is based on Eclipse (earlier Netbeans) or Flash*. Thus CTAT relies on a project structure which is largely organized to function within the development environment: source-files are together, static resources are separate, ... A screenshot of the project files with a command to "run" a given source extracted from the CTAT documentation is on the left.[1]

[1]This screenshot is extracted from the section *Configuring NetBeans 5.0+* at `http://ctat.pact.cs.cmu.edu/index.php?id=netbeans50`.



Illustrator's and the dynamic geometry systems' elementary files are standalone but both export to multiple files and directories when publishing to the Web: the picture on the left shows the icon of an Illustrator file from the same combinatorics textbook and a file contributed to i2geo* by a Spanish teacher.[2]


AHA! Application Management Tool
File Edit Operation Tools Help
Application Files
Author Files
Info

D:\aha\tutorial

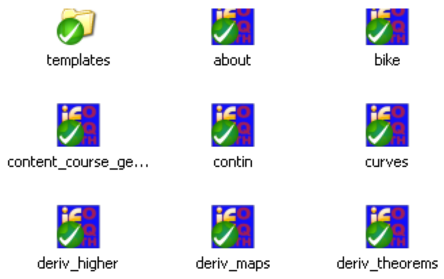
Local
CVS
icons
images
xml
ConceptTypeConfig.xml
footer.xml
header.xml
index.html
LayoutConfig.xml
Registration.html

/tutorial

CVS
icons
images
xml
ConceptTypeConfig.xml
footer.xml
header.xml
index.html
LayoutConfig.xml
Registration.html

Action	Source File	Source Directory	Destination Directory	Status

AHA! authoring is on the web, launched through the signed applet* s hosted on the AHA! server. As explained in the tutorial the authoring starts at the *Application Management Tool*[3] and, precisely, that tool presents two file-systems: the server's and the client's with methods to upload and download.



jEditOQMath is focussed on the edition of the source files, the `.oqmath` files. As described in Chapter 6, the files are packaged in content-collections. A part of the directory of OQMath sources of LeAM_calculus* is shown on the right; it is a file-manager view on the Windows operating system decorated by the TortoiseSVN plugin displaying the checkmarks that indicate that files are up-to-date.

The concretization of files allows three important activities of the authors:

- the ability to perform versioning, backup, and copy activities,
- be archivable (e.g. in zip files) and exchangeable over the web or email
- be the entry point to start editing or to deploy

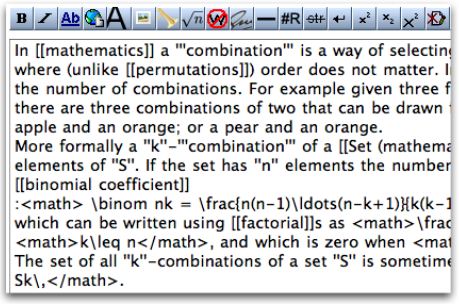
1.4.3 Authoring Sources Edition

The authoring's central work is the manipulation of authoring tools which, by themselves, manipulate a set of data: the space of *authoring sources* is where this data is edited and viewed. The information in authoring sources is only partially relevant to the functions of the learning-environment activity: the learning-environment or a build-system distill the only necessary facets, the rest being left as help for authoring.

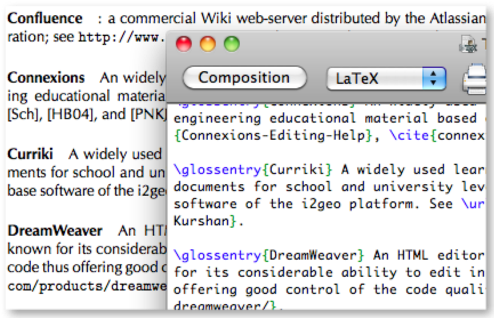
[2]See this resource at `http://i2geo.net/xwiki/bin/view/Coll_mabanades/PascalsLimacon`.

[3]The tutorial is that of AHA! 3.l0 at `http://aha.win.tue.nl/` where one needs to register. It is delivered with each version of AHA! as the basic example of adaptivity.

MediaWiki authoring sources are made of the wiki plain-text source and the attachment data. They are often edited in web-browsers or in text-editors with dedicated syntaxes. The figure on the right displays editing the page on *Combination* of the English WikiPedia[4] edited in a web-browser; the source contains text, links, and formulæ. Buttons in the toolbar create *templates* for typical fragments to be inserted.



```
In [[mathematics]] a '''combination''' is a way of selectin
where (unlike [[permutations]]) order does not matter. I
the number of combinations. For example given three f
there are three combinations of two that can be drawn 
apple and an orange; or a pear and an orange.
More formally a ''k''-'''combination''' of a [[Set (mathema
elements of ''S''. If the set has ''n'' elements the number
[[binomial coefficient]]
:<math> \binom nk = \frac{n(n-1)\ldots(n-k+1)}{k(k-1
which can be written using [[factorial]]s as <math>\frac
<math>k\leq n</math>, and which is zero when <mat
The set of all ''k''-combinations of a set ''S'' is sometime
Sk\,</math>.
```



Confluence : a commercial Wiki web-server distributed by the Atlassian
ration; see http://www.

Connexions An widely
ing educational materia
[Sch], [HB04], and [PNK]

Curriki A widely used
ments for school and un
base software of the i2ge

DreamWeaver An HT
known for its considerab
code thus offering good c
com/products/dreamwe

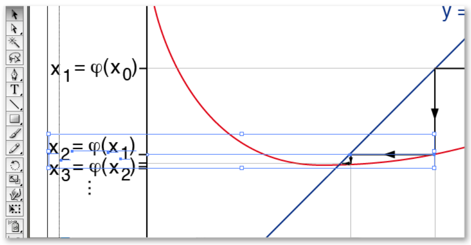
```
engineering educational material based
{Connexions-Editing-Help}, \cite{connex

\glossentry{Curriki} A widely used lear
documents for school and university lev
software of the i2geo platform. See \ur
Kurshan}.

\glossentry{DreamWeaver} An HTML editor
for its considerable ability to edit in
offering good control of the code quali
dreamweaver/}.
```

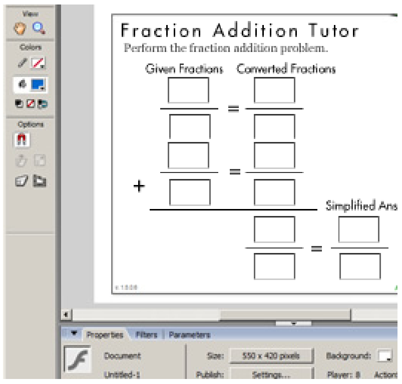
TeX sources are made of the plain-text source files and its included files. Using a dedicated TeX authoring tool supports the authors by such features as syntax-highlighting or running and previewing the result. The screenshot on the left displays the TeXshop application[5] in action, offering support for the typesetting of Chapter 6 of this thesis. The source is in the front and the preview at the back.

Adobe Illustrator edits a view that is expected to apply exactly the *what you see is what you get* paradigm. Nonetheless, multiple tools and enrichments of the view such as the selection display, can make this paradigm weaker. One should still consider source editing because only the *native* formats (currently, `.ai`, `.eps`, and `.pdf`) support the storage of all features of the editor. Exporting to other formats may loose the vector information, or simply the groups or layers. The screenshot on the right is in the edition of a graph picture of LeAM_calculus*.

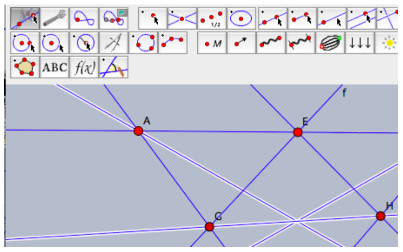


[4]The English page about *combination* can be reached at http://en.wikipedia.org/wiki/Combination.

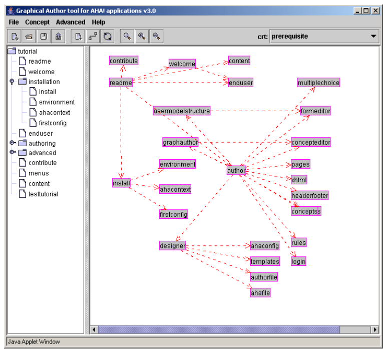
[5]TeXshop is an open-source TeX-authoring environment for MacOSX, see http://www.uoregon.edu/~koch/texshop/.



CTAT editing, both for Flash* and Eclipse, is the combination of the editing tasks for the visual component construction tools, which is close to WYISWYG but is complemented by a very rich set of property editors and the editing task of the behaviour graph, which shows different stages of the user-interface when it is running. The source is thus both the visual components construction and the multiple property values for each of the behaviour graph vertices. The picture on the left shows the visual editor of Flash* when designing a fraction addition tutor, it is extracted from the CTAT tutorial.[6]



Interactive geometry software files are generally specific to the software they are made for. This specificity supports particular design choices (some like it with a touch of a cartesian frame, some don't, some like it web-ready, some don't...). Editing the *source* files involves, mostly, manipulating the geometric configuration made of geometric elements (points, lines, segments, circles, conics, intersections, labels...) which remains free for some of them, while keeping the geometric constraints. The authoring process is typically done within a dynamic geometry system that has all the functions to edit while the learning resources made available to learners try to contain only the necessary functions. The picture on the right is a remake of a Cinderella construction demonstrated by U. Kortenkamp as an introduction to interactive geometry.[7]

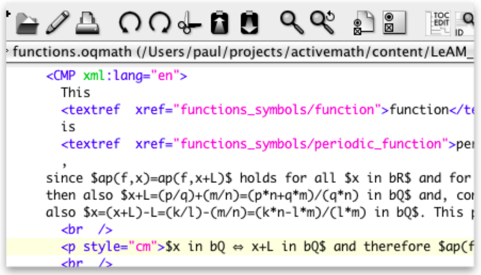


In AHA!, one edits sources in several ways: the domain model is edited in the Graph Authoring Tool (a visual editor) or in the Concept Editor (a form-based editor which also allows the edition of the adaptation model), the content files are typically edited externally (in an XHTML editor with all the page-internal adaptivity works provided by dedicated tags which require hand editing the XHTML). Finally, user-model-input can be supported by an enriched source authoring called the Form Editor; see [BSS$^+$07] for details on these tools.

[6]The CTAT tutorials are available at `http://ctat.pact.cs.cmu.edu/tutorials/`; this screenshot is extracted from *Building a student interface for fraction addition (Flash)*.

[7]This talk is available at `http://vimeo.com/3826066`.

All of these objects are hand-editable as XML* files. The image on the left is from the AHA! tutorial mentioned above.



```
functions.oqmath (/Users/paul/projects/activemath/content/LeAM_
      <CMP xml:lang="en">
        This
        <textref  xref="functions_symbols/function">function</te
        is
        <textref  xref="functions_symbols/periodic_function">per
         ,
      since $ap(f,x)=ap(f,x+L)$ holds for all $x in bR$ and for
      then also $x+L=(p/q)+(m/n)=(p*n+q*m)/(q*n) in bQ$ and, cor
      also $x=(x+L)-L=(k/l)-(m/n)=(k*n-l*m)/(l*m) in bQ$. This p
        <br  />
        <p style="cm">$x in bQ ⇔ x+L in bQ$ and therefore $ap(f
        <br  />
```

jEditOQMath is focussed on editing the OQMath files of the content collections. These are XML*-files that are kept in readable plain text. Editing involves modifying the sources, either using facilitating tools (predefined templates or copy-and-paste) or using simple text manipulations; it also involves validating the correct encoding. The image on the right a is a typical image that an author at work for LeAM_calculus* might have seen.

Authoring sources are the places where the author can edit and add further content. They provide one view of the content which is different than the view the recipients will have. Authoring sources only make sense if they *can produce content*, that is, only if a transformation from the editing spaces to the learning environment, called *deploy* or *export*, can be done.

Authoring sources are the central place of authoring; hence they tend to be the result of precious tweaks so that authors find their content in a way they are used to. They need to be presented with ways to organize themselves and using familiar paradigms.

Authors that work towards the same project can only share a common set of authoring-sources; depending on the role distribution, this can mean:

- to use the same authoring tool and to share the common files and the same encoding practices,
- or that different people use different tools and edit different parts of the content. This is done in organized teams who distribute the work in a coordinated way among the various specialists, for example in [Zim08].

Authoring sources are not only made of words or graphics that will be shown to the learner. They are very often also made of more abstract content such as the knowledge used for the intelligent behaviours of the learning environment (e.g. the metadata* of each ActiveMath item in jEditOQMath, the behaviour graph for CTAT), the hyperlinks allowing a learner to navigate (present in all systems, input as a graph or as text), or the ways to constrain and evaluate the learner's actions (constraint in dynamic geometry systems, interactive exercise input evaluations in CTAT and ActiveMath). The meta-model of [Mur03] describes a model of the organization of knowledge which applies to most intelligent learning environments; most of the ingredients there are *the source* but almost none are actually made of content presented to learners.

1.4.4 Build and Preview

The authoring work's goals are realized within a consuming environment - in our case, a learning environment. Having edited a significant amount of content sources, the author will wish to see the result in the learning environment. This is done in two phases: the build-process and the learning environment preview.

The authors access the content here by commanding the build process and perceiving the error report of the build tool.

The preview set-up is the usage of a dedicated instance of the learning environment in order to obtain a view that is close to what the learners will be given when in production. It would be an ideal situation to make this process an automated consequence of the build-system, this would be, however, ignoring the multiple facets of the usage of the learning environment: for example the display of a piece of content as a single entity in a search tool or within a larger context in a book (in ActiveMath or Illustrator for example); or the publication in different media forms (paper or web forms for example).

The write-build-preview cycle is a classical paradigm of any authoring work, not only e-learning; it is implemented in most editing tools and is fundamentally justified by the fact that the result of the authoring work needs to be evaluated in a setting as close as possible to that of the consumers. It has been my experience that only a realistic preview which is very close to what the learners will be delivered is able to attract the trust of the authors and that no features of the authoring sources are useful if they cannot be proofed within the target environment.

Write and preview cycles appear in [Sha99] to be the fundamental approach to successful writing *as creative design*.

Build and preview cycles can be seen in some fashion within each of the authoring environments we consider. We do not include pictures of these features in this section since the *build* process is more a command than a visual state:

- MediaWiki's editing forms allow the save and preview buttons to see the web-page that will be delivered. These actions are often followed by tests, and subsequent edits, e.g. to adjust an improperly encoded bit of syntax, to obtain the right formatting, or to obtain the right link.

- any TeX environment is endowed with a compile and preview action; for the very popular LaTeX workflow, it is particularly necessary since it is the only way to observe the resulting layout as well as such derived content as a table-of-contents or an index.

- Illustrator's build cycles are not so present because the visual target is made immediately visible. In most cases, however, the work on Illustrator aims at derivatives which need preview cycles: for example the insertion within a wider printed material (where one needs to verify that colours match to their environments, that they print well, ...), the insertion as picture in a presentation (where readability is at risk), the insertion within a web-page... In all these cases, the preview cycles appear and are, sometimes, expensive.

- CTAT relies on Eclipse or Flash* which both have several forms of build and test environments. CTAT's paradigm, called *programming by demonstration*, uses previews as an editing interface (where the author is in *demonstrate* mode) but the authors are also invited to test their tutors for which CTAT offers the *test tutor* mode. Going from one to the other does involve a form of build which the IDE routinely executes. Finally, an optimized build is necessary to produce the necessary files to be run at the learners' machines.

- Dynamic Geometry Systems are authoring tools most of the time. What is delivered to the learner, in general, is a pre-filled construction with some elements that are fixed. Generally only the result of the export allows an author to see the construction with all the restrictions a learner will have.

- AHA!'s preview cycles don't seem to be made explicit but the *AHA! application* is immediately visible to a user hence testable; clearly however modifications to the sources are followed by uploads and various choices which can be seen a part of the build process.

- jEditOQMath is based on the preview cycles, which we call WYCIWYG* in Chapter 3: the preview in the (author-local) ActiveMath learning environment is the only way to see the combination of the source edited in jEditOQMath with the other content elements (such as the pictures or content items that are re-used) and to see the content as it will be delivered to the learner. ActiveMath's rich features, which include individual sequencing, search tools, and multilinguality, makes it necessary for an author to preview the authoring result in several expected usages of the content.

One distinguished feature of authoring tools is how the **preview cycles** can be **made shorter**. This is key to support explorations of authors that discover at the same time the learning environment and the authoring capabilities; we believe that this is a fundamental aspect of achieving the desire of authors to act alone and exploratively.

- MediaWiki allows section editing which is a simple way to concentrate the editing cycles to a smaller part. It is sometimes difficult, however, to jump from a part in preview to a relevant part in source.

- TeX's classical *compile from text* approach, the same as that of programmers, allows it to be easily integrated in text-editors workflows where the compilation results can contain error reports which can take the user to the exact place reported about. Only some TeX environments allow the navigation from source to preview locations (and back). TeXshop is one such example: a mouse-click combined with the press of the apple-key (or command-key) lets the author go from a word in the preview to that word in the source and back again.
- A few specific *pre-flight-check* features can be exploited in Illustrator (for example to check that the black colour is always on top of layers and is a bit broader to cope for printing imprecision). They support the jump into the source at faulty places.
- CTAT allows the change of mode anywhere in the behaviour graph. It is thus easy to move within the graph by using and adjusting the parts as they appear. However not all changes are kept and one needs to re-build to make sure of the facets of what is delivered to the learners.
- jEditOQMath only allows authors to go from preview to source by offering a link next to each content-item that is only accessible to users that are registered authors. However jEditOQMath streamlines the build and reload cycle so that redoing a build is a single button press with the preview obtained by a browser reload.

It should be noted that previewing may be for the purposes of actually testing the *invisible content*, such as that of the metadata* or the domain model. For example, one may preview content through the usage of a search tool and revise that content and proof it again in the same tool.

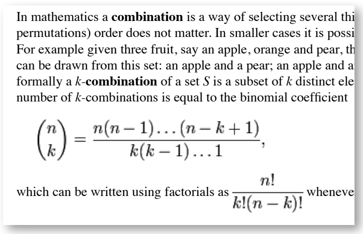
1.4.5 Content in the Learning Environment

The objective of authoring is to have the content usable in the learning environment. The content is stored inside a server software and a part of it, that corresponds to the current learning objective, is presented to the learner. In this space, the author does not see it in a form that represents the internal storage but in a form that represents what he expects the learners will see.

In web-delivered learning environments, much more than just the author's own learning content can be seen this way: the author can connect to the environment of others and enjoy it before actually starting a re-use action (see Chapter 6).

In our sample authoring tools:

MediaWiki offers access to the content as pages with links among them or access through the search tool.

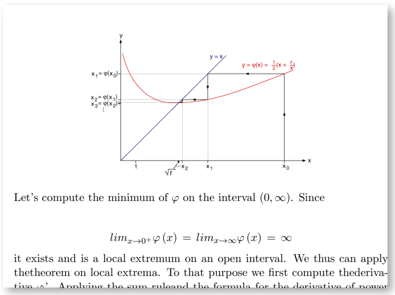


In mathematics a **combination** is a way of selecting several thi
permutations) order does not matter. In smaller cases it is possi
For example given three fruit, say an apple, orange and pear, th
can be drawn from this set: an apple and a pear; an apple and a
formally a k-**combination** of a set S is a subset of k distinct ele
number of k-combinations is equal to the binomial coefficient

$$\binom{n}{k} = \frac{n(n-1)\ldots(n-k+1)}{k(k-1)\ldots 1},$$

which can be written using factorials as $\frac{n!}{k!(n-k)!}$ wheneve

TeX's results, nowadays, most often results in PDF* files which can be web-delivered, e.g. from a course web-page. A picture here is provided by this thesis.



Let's compute the minimum of φ on the interval $(0, \infty)$. Since

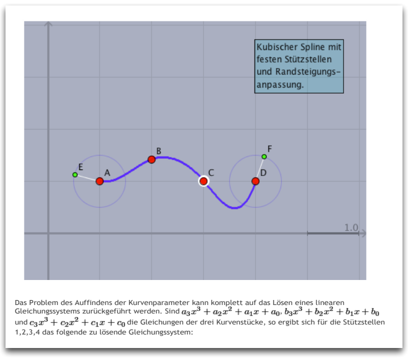
$$lim_{x\to 0+}\varphi(x) = lim_{x\to\infty}\varphi(x) = \infty$$

it exists and is a local extremum on an open interval. We thus can apply
thetheorem on local extrema. To that purpose we first compute thederiva-

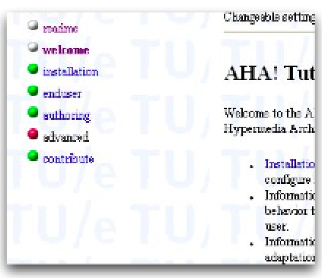
Illustrator's pictures are not particularly designed for a delivery environment but their output in other formats is common. In the picture on the left, we see the Illustrator picture of LeAM_calculus* mentioned above within one of its delivery targets: the print output by ActiveMath of this book's page. Among the features to proof here are the consistency of formula between the picture and its environment (e.g. the consistency and readability of notations).

CTAT's tutors are currently delivered as simple links to applet* s (be they in Flash* or Java*). A click on each starts the tutor which constitutes an interactive exercise. A view very similar to the view of CTAT in section 1.4.3 would apply here.

Dynamic geometry systems, similarly, deliver their resources within simple web-pages which embed the applet* s. The picture on the right is an extract of MatheVital, one of the exemplary web-sites that explains chosen mathematical topics supported by the explorations with the dynamic geometry figures.


Kubischer Spline mit festen Stützstellen und Randsteigungs-anpassung.
A
B
C
D
E
F
1.0

Das Problem des Auffindens der Kurvenparameter kann komplett auf das Lösen eines linearen Gleichungssystems zurückgeführt werden. Sind $a_3x^3 + a_2x^2 + a_1x + a_0$, $b_3x^3 + b_2x^2 + b_1x + b_0$ und $c_3x^3 + c_2x^2 + c_1x + c_0$ die Gleichungen der drei Kurvenstücke, so ergibt sich für die Stützstellen 1,2,3,4 das folgende zu lösende Gleichungssystem:



AHA! Tut

Welcome to the A
Hypermedia Arch

- Installatio
configure
- Informatio
behavior t
user.
- Informatio
adaptation

AHA! builds a web-application which one can install as web-service. They have their own navigation which is the essence of adaptive navigation. Generally, a table-of-contents is displayed with parts inviting to be read and parts discouraged to be read. The picture on the left (extracted from [BSS+07]) describes this feature, showing some table-of-contents green (which means *go*) and some red (*stop*).

ActiveMath content is accessed in the form of books that are listed in the main-menu, through the search tool, or through personal books which the course generator produces on demand (see Chapter 2).

Accessing the content in the learning environment is, for the author, the main chance to test the learning environment's features concerning the content of interest. Only based on his explorations will he be able to write precise instructions for his target learners and will he be able to trust the feasibility of a navigation action he expects of them.

While being technically the same, the content access in the learning environment can have at least three different **purposes**. Strictly speaking, these could be three different spaces.

- The authoring preview is the most elementary form; there, the author manipulates and sees it with the objective of evaluating the result of his creation process and see how he can refine the content.
- The learning environment used by the target learners, to where editors transfer the content made available. If at all, authors or editors, see the result of the learning in the form of reports on progress of learners (or in the form of teaching feedback). Authors and editors may have the important role of explaining how the content is best used.
- The learning environment made available to the public so that others can see the content in action to evaluate their characteristics; for example, to evalute re-using content.

1.4.6 Published Content Sources

A final space where content appears is the sharing place which allows authors to transmit the authoring source files, for example, for re-use. The storage can be in the form of published archives, of access to versioning systems, or as web-service. The web makes it easy to access a large number of such archives.

Generally, the publication of content sources is organized in a way that is disjointed from the learning environment; that is indeed the case for all seven tools considered above.

Web-based communication places are among the best places to share the information about published content sources. These communication places, such as Connexions*, Curriki*, or i2geo*, are used to describe what others have considered good or bad, or in which situation one can use the content. We describe in section 6.8 how this communication between authors can be concretized in more detail.

1.5 Conclusions

In this Chapter, we have attempted to define the instruments that authors use when creating and managing content. The aim is to classify the places of interaction with the content which are far more diverse than the mere editing tools.

We have called the instruments *content spaces* because they represent places where the content is living, where the author dives in to manipulate it. Although the content is one and the same, it appears fundamentally differently in different spaces.

This Chapter has attempted to differentiate the authoring from that of a developer, illustrating it by several classical tools. We claim that this perspective on the authoring tools makes a useful model to bring the authoring tools descriptions to the more realistic land of authors who learn the art of authoring and the art of manipulating the learning environment at the same time.

To conclude, we make an analogy between the authoring work and the work of a sculptor:

- Non-computerized-places probably correspond to the inspiration sources which a sculptor may use.
- External manipulations of content containers: the time when the sculptor lays materials and tools out before beginning work with them.
- Edition of content sources: the time when the sculptor actually sculpts the content. Depending on the material, different tools are used.
- Build of content: corresponds to all the transformation processes a sculptor would be working with in order to obtain his effects (e.g. painting, bathing...).
- Testing the content in the learning environment: the time when the sculptor enjoys the result of his sculptures under very many possible lighting situations and maybe even by touching them in many possible ways.
- Deployment of the content: the time when the sculptor installs the sculptures in an exhibition area.
- Published content sources: the time where success comes that reproductions are being made or offered to order (e.g. in catalogs).

Chapter 2

ActiveMath and Semantic Mathematical Documents

Introduction

ActiveMath is a web-based learning environment for mathematics. It supports the learning process by a quality content presentation combined with intelligent services (such as the interactive exercises or the course generator) and the usage of learning tools. It is an adaptive system which recommends the learning content deemed appropriate to the learner, based on the its estimate of the learner's competencies.

ActiveMath has grown in research projects from around the year 2000; it has been developed by a team of workers, mostly assuming the dual role of researcher and coder, under the supervision of Erica Melis. ActiveMath has been often evaluated in real classroom settings, from 6th-graders (13 y. old) until the University undergraduate level. We refer to Chapter 9 for more details of its evaluations.

Although I have taken part in the development of some of the parts of the ActiveMath learning environment (as can be seen in the publications' authorship), I have included in this thesis only the aspects relevant to authoring.

This thesis describes how to realize *content* for the ActiveMath learning environment: that is, how to create a learning experience that learners work through. The authoring activity creates learning *content* that is loaded by the server software where it is displayed, evaluated, recommended, or filtered. In this chapter we introduce how the content is organized, i.e., we describe the knowledge representation, then we describe how it is used in ActiveMath to produce the learning experience.

The learning content is made of three levels which we shall describe separately in the sections below: the formulæ, the documents, and the metadata*. Mainly, this knowledge representation has been contributed to by third parties which we refer to. After this description, we provide an overview of the ActiveMath concepts as well as its software components and tools; for some, we refer to other chapters.

Readers of this chapter should understand the magnitude of the mission of the mission to create the content of ActiveMath. This sets forth the main mission of users of the authoring tool set.

2.1 Semantic Mathematical Formulæ

At the smallest level of the content of ActiveMath are the mathematical formulæ: they are encoded using the OpenMath language, an initiative to define an interoperable standard for encoding mathematical formulæ by their meaning. We call the OpenMath expressions *semantic mathematical formulæ* as opposed to the expressions that are written mostly to render appropriately such as the TeX* or MathML*-presentation encodings. OpenMath has emerged around 2000 from a group of researchers in computer algebra and automated theorem proving.

Its aim was to be an exchange language between multiple mathematical systems. Our work is based mostly on OpenMath 2 [BCC$^+$04]. OpenMath expressions are built as XML* trees where the elements are one of:

- **variables** (`OMA` elements) which bear a name
- **symbols** (`OMS` elements) which point to a content-dictionary entry where its semantics is declared)
- **constants** (such as `OMF` or `OMI`)
- **applications** which *apply* a symbol to the arguments (`OMA` elements with the first child being the mapping, the others being the arguments)
- **bindings** or **attributions** which are a special form of applications that limit the scope of a variable to its binding (for example with a $\forall$ or $\exists$ symbols)

```
<OMOBJ>
  <OMA>
    <OMS cd="relation1" name="eq" />
    <OMA>
      <OMS cd="transc1" name="sin" />
      <OMA>
        <OMS cd="arith1" name="plus" />
        <OMS cd="nums1" name="pi" />
        <OMV name="α" />
      </OMA>
    </OMA>
    <OMA>
      <OMS cd="arith1" name="minus" />
      <OMA>
        <OMS cd="transc1" name="sin" />
        <OMV name="α" />
      </OMA>
    </OMA>
  </OMA>
</OMOBJ>
```

An example of an OpenMath expression is given on the left. it expresses the relation:

$$\sin(\pi + \alpha) = -\sin\alpha$$

In this expression, one sees the variable elements, the application elements and the symbol elements. The symbol elements bear attributes `cd` and `name` which indicate the symbol's content-dictionary (the document where the symbol is declared) and the symbol's location inside it. For the sine function mentioned here, the symbol is declared in the content dictionary `http://www.openmath.org/cd/transc1`. Its declaration can be browsed at `http://www.openmath.org/cd/transc1.xhtml#sin`: the web page at this URL is a rendering of the content dictionary entry which provides a description of it and some of its mathematical properties in the form of equations written in OpenMath. The fact that the content-dictionary has this URI follows from the `cd` attribute as well as the `cdbase` attribute in its environment which defaults to `http://www.openmath.org/cd/`.

In the authoring system presented in this thesis and in the ActiveMath learning environment, the formulæ in OpenMath are produced by an author who enters a compact linear syntax called QMath or by learners using one of the input facilities. Several things can then happen to the formulæ – they can be:

- stored and *resolved* by the content storage,
- rendered to one of the browser languages by the presentation pipeline,
- computed by the computer algebra systems,
- graphed by the function plotter,
- compared by the exercise system,
- or matched by the search engine.

Although the OpenMath expressions are very verbose, they tend to be easy to conceive and understand. As we shall show in other chapters, the authors will not see OpenMath in their everyday activity but will understand it behind multiple representations of it.

2.2 Semantic Mathematical Documents

For mathematical content to be written, more than simple formulæ must be expressed. In particular, mathematical documents need to include fragments of texts interleaved with mathematical formulæ. Also, if interactive exercises are desired, another part of the language used in mathematical documents should express how they are to behave.

The OMDoc language emerged for such a mission. It goes far beyond the bare content dictionary, which simply gathers a set of symbols with their properties, OMDoc proposes a semantic organization of mathematical texts that include textual and formal fragments each with a mathematical role: OMDoc was created by Michael Kohlhase around 2000. Versions 1.1 and 1.2 are the ones that have been mostly used in ActiveMath with some exceptions that we describe below. OMDoc 1.2 is documented in [Koh06]. When indicating an element name that is new or different than OMDoc in ActiveMath, we shall use the + exponent, e.g., `textref`$^+$.

The OMDoc language is again an XML* language. For the purposes of ActiveMath, OMDoc is made of the following elements:

- structural elements: those are the elements that allow the content organization within the XML documents.
  - `omdoc`: a container for anything inside OMDoc documents.
  - `theory`: also a generic container which, importantly, contains an identifier; one considers that OMDoc is made of mathematical theories, each in a `theory` element, which may relate to each other.
  - `imports`: this element, a child of `theory`, allows content items in other theories to be referenced to using a short string. The reference resolution mechanism is explained below.
  - `private` elements are used to insert data which are, more or less, for a special purpose; among the widespread usages, the `private` and `data` elements are used to instruct the embedding of pictures or applet* s within the rendering of OMDoc documents in HTML* and PDF*.
  - `ref` and `textref`$^+$ are elements inside the content which allow a reference from a content item to another. They can use relative or absolute references. `ref` is of generic nature, used in multiple places, while `textref` is expected to mean a hyperlink from a text fragment to a content-item. Both contain the reference in the `xref`$^+$ attribute (contrary to OMDoc which stipulates that this attribute is for *structure sharing*: it can be inserted in almost any element to include the referenced element in place of the the element).
- The content items in OMDoc are the atomic carriers of content. Each contains an `id` attribute which allows it to be referenced.
  - `symbol` is an element defining a conceptual entity which is typical of mathematical symbol. It contains a title and a description (which, contrary to OpenMath* content dictionaries, can contain formulæ and be in several languages).
  - Conceptual content items: `definition`, `axiom`, `assertion`, `proof`, and `method`$^+$ are all normal parts of a mathematical document with a dedicated role. These content items carry the *essential* part of the mathematical knowledge contained in the documents.
  - Satellite content items: `example`, `exercise` and `omtext` are content items which are less central. Each of these content items need a reference to a conceptual content item in the attribute `for`.
- The micro-structure: within the content items lies the annotations and the text fragments which make the substance of the content. They are organized in one `metadata` element (the annotations) and a series of `CMP` (the text).

- `metadata` and, inside it, `extradata` contain the annotations. Bibliographical information appears there. Most pedagogical annotations, which have been defined by the ActiveMath group, come within the `extradata` element, which OMDoc depicts as a particular extension point.
- `CMP` is the elementary paragraph element; its content forms the text of the content element. It supports an attribute `xml:lang` to indicate the language thus allowing multilingual content items.
- `with` enriches its content with a particular style.
- `omlet` inserts an image or applet* by reference to a `private` element.
- `OMOBJ` encloses a mathematical formula expressed as an OpenMath* element.
- several other elements are available. We refer to the DTD* which is distributed with each version for a complete authoritative list and a few indications, as well as the OMDoc 1.2 book [Koh06] for extensive explanations, and to [Gog11] for the elements relevant to the exercise sub-system of ActiveMath.

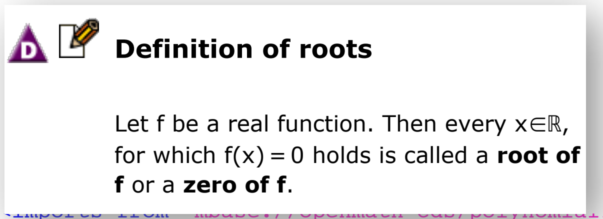
An example of an OMDoc fragment is in Figure 2.1. It is a very simple definition – that of the concept of the root of a function: this concept is represented by the `symbol` element having the identifier `root`. The definition requires another concept, that of the notion of function, and contains several formulæ in OpenMath* which contain references to other symbols (e.g., the set of real numbers $\mathbb{R}$ or the set membership $\in$).

The information organization of an OMDoc document aims at semantic based interoperability: because the fragments have a precise role, because the formulæ are encoded in a computable fashion, and because of the metadata* information, one expects multiple services to be able to exploit the information contained in OMDocs. This interoperability is expected by the meaning-oriented encoding as opposed to a pure presentation oriented encoding (such as TeX*). Among others, rendering in multiple forms and media should be achievable from OMDoc. Computations or plotting should be achievable from some of the formulae, and searching for formulae is likely to be more precise.

2.2.1 Identification and References across OMDoc Documents

For the services of the ActiveMath learning environment, it is central to label each of the content items with an identifier and to reference these from other places – e.g., hyperlinking or annotating a relationship. OMDoc requires each of the content items and each of the theories to have an identifier, within the *id* attribute.

These strings are expected to be easily recognized by authors and to contain only simple characters. They can be inserted in the `xref`$^+$, and `for` attributes of other


Definition of roots

Let f be a real function. Then every x∈ℝ, for which f(x) = 0 holds is called a **root of f** or a **zero of f**.

```
  <imports from="mbase://openmath-cds/polynomial1" />
  <imports from="mbase://openmath-cds/quant1" />
  <imports from="mbase://openmath-cds/relation1" />
  <imports from="mbase://openmath-cds/set1" />
  <imports from="mbase://openmath-cds/setname1" />
  <imports from="mbase://openmath-cds/transc1" />
</omgroup>
  <symbol id="root">
    <commonname>root</commonname>
  </symbol>

  <symbol id="function">
    <commonname>function</commonname>
  </symbol>

  <definition id="def_root" for="root">
  <metadata>
    <Title>Definition of roots</Title>
    <extradata>
      <relation  type="domain_prerequisite" >
        <ref  xref="function" />
      </relation>
    </extradata>
  </metadata>
  <CMP>
    Let <OMOBJ><OMV name="f"/></OMOBJ> be a real
    <textref xref="function">function</textref>.
    Then every <OMOBJ>
      <OMA><OMS cd="set1" name="in" />
        <OMV name="x" />
        <OMS cd="setname1" name="R" /></OMA></OMOBJ>,
    for which <OMOBJ><OMA><OMS cd="relation1" name="eq" />
        <OMA><OMV name="f" /><OMV name="x" /></OMA>
        <OMI>0</OMI></OMA></OMOBJ>
    holds, is called a <highlight>root</highlight>
    of <OMOBJ><OMV name="f"/></OMOBJ>
    or a <highlight>zero</highlight>
    of <OMOBJ><OMV name="f" /></OMOBJ>.
  </CMP>
</definition>
```

Figure 2.1: An example OMDoc document extracted, made monolingual, and slighly simplified from the LeAM_calculus content collection (see Section 9.3.2 about its author). The `definition` element is presented together with relevant `symbol`s with its rendering in ActiveMath on top

OMDoc elements, where they can denote a reference. They can also be inserted in the attributes of the `OMS` OpenMath* symbol elements, which reference an identifier (the `name` attribute), a theory (the `cd` attribute), and a collection (`cdbase` attribute).

One can reference using the same string as the identifier. This is the simplest reference. For example, the `for` attribute of the `definition` element in Figure 2.1 is `for="root"` in order to indicate a reference to this symbol in this same `theory`.

Writing references beyond the sibling element as done above can be done in OMDoc 1.2 but has been almost only using file references. In contrast OMDoc aims at supporting `theory` and `imports` elements suggested by the development graph approach [MAH06b]. This approach stipulates that theories can be imported in to other theories, and this has the effect that content items of the imported theory can be referenced as content items of the theory containing the import: with one short identifier. This apparent inconsistency, that was due to the lack of indexed storage in the OMDoc world, has stimulated ActiveMath to adopt another reference scheme which supports the theories and imports.

This reference scheme, the `mbase://`-scheme, is explained in Section 4.4.2 in full detail. Suffices to say that it is based on **content collections**, which are directories corresponding to authoring projects, each containing a series of **OMDoc files** with **theories**.

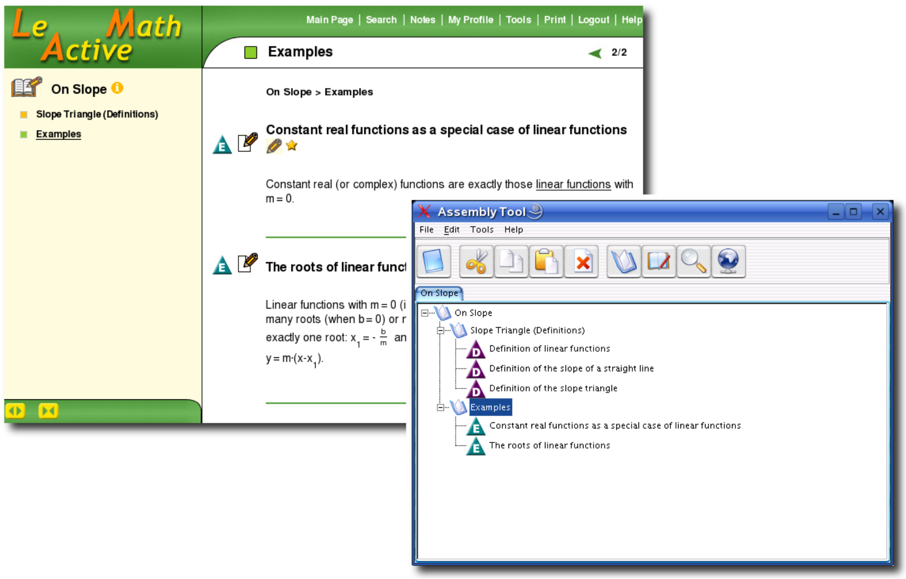
ActiveMath makes extensive use of the references, notably within the pedagogical annotations which are the basis of the student modelling and course-generation features (e.g., to say that a definition requires a given concept as the domain_prerequisite relation in Figure 2.1).

ActiveMath also uses references in the *table of contents*: they are hierarchies rooted in an `omgroup` element which can contain further `omgroups` (the chapters and sections) as well the `ref` element which refers to a content item. Any `omgroup` that contains a child being a `ref` is a page for ActiveMath, while the enclosing `omgroup` represents a book which can be inserted in the main menu. Pages or chapters can be enriched with titles. Such a hierarchy allows ActiveMath to display the book content by title in a hierarchically organized way and learners to edit their books using the assembly tool as seen in the Figure 2.2.

2.2.2 Annotations of the Content Items of OMDoc

OMDoc content items are atomic elements of texts. They carry the information of an identifier, a type, textual content, and annotations. The annotations are all contained within the `metadata` element. They provide additional information which serves multiple purposes.

Among the early objectives of OMDoc was a wish to contain annotations of bibliographical nature: a title (useful to display to the users), an author, a version - all of these being aimed at the management of the item. These annotations are taken


Le Math Active
Main Page | Search | Notes | My Profile | Tools | Print | Logout | Help
Examples
2/2
On Slope
Slope Triangle (Definitions)
Examples
Assembly Tool
File Edit Tools Help
On Slope

On Slope > Examples

Constant real functions as a special case of linear functions

Constant real (or complex) functions are exactly those linear functions with m = 0.

The roots of linear funct

Linear functions with m = 0 (i
many roots (when b = 0) or r
exactly one root: $x_1 = -\frac{b}{m}$ an
$y = m\cdot(x-x_1)$.

On Slope
Slope Triangle (Definitions)
Definition of linear functions
Definition of the slope of a straight line
Definition of the slope triangle
Examples
Constant real functions as a special case of linear functions
The roots of linear functions

Figure 2.2: A typical configuration of the Assembly tool which learners and teachers can exploit to edit the table-of-contents of books of ActiveMath.

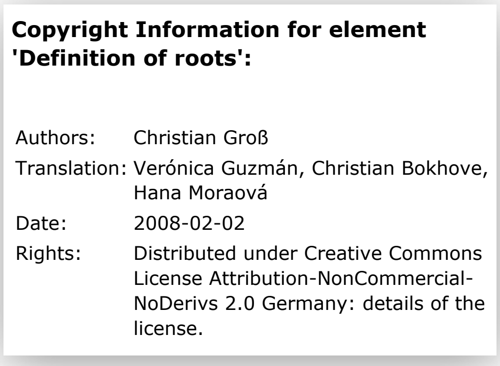
from the widespread Dublin-Core* metadata standard. The metadata element is also the carrier of pedagogical annotations which contain the essential knowledge used by the services of ActiveMath.

The bibliographic metadata is described in details in the book [Koh06] while the pedagogical metadata is described in an internal document called the metadata specification. Two public versions of it have appeared – [GUM$^+$04] and [Sos10]. Our description is mostly that of the first. For working authors, the details of the accepted values for the metadata are specified in the DTD* that their *content collection* is using; the authoring tool validates the allowed values based on the DTD and suggests possible values.

Thus it has been possible to elaborate a particular vocabulary of pedagogical annotations for the purposes of a specific project which would require a set of custom vocabulary values.[1]

The annotations carry the information as properties of the content item: a property is carried by its name and its attributes (the language, the level, ...) and has a value.

[1]This practice has been quite common. Among others, the Schulmathematik and Matheführerschein projects used it. They are typical of projects which involved the ActiveMath group to create customizations of the learning environment.



Copyright Information for element 'Definition of roots':

Authors:	Christian Groß
Translation:	Verónica Guzmán, Christian Bokhove, Hana Moraová
Date:	2008-02-02
Rights:	Distributed under Creative Commons License Attribution-NonCommercial-NoDerivs 2.0 Germany: details of the license.

Bibliographical information properties can be simple strings (e.g., `Title`, `Creator`, `Contributor`, `Publisher`) or date (`Date`), can be specific to a role (e.g., the contributor is a translator), and can also be specific to a language (e.g., the contributor is a translator to French). Except for the title, which is displayed every time the content item is shown, this information is displayed by ActiveMath in the *copyright information*, a sample of which is on the right.

Pedagogical properties have values either within an enumerated set of possibilities or are relations which reference another content item:

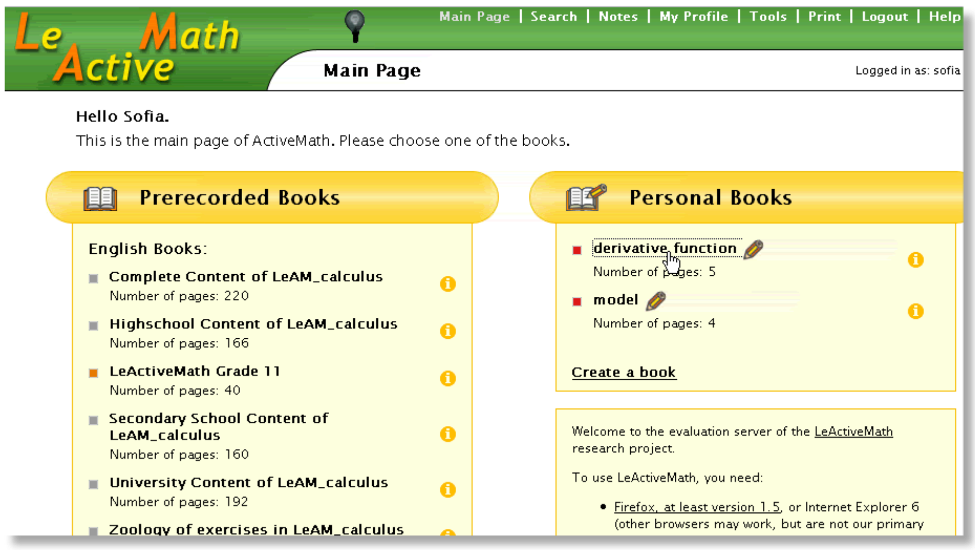
- Pedagogical properties with enumerated values include `learningcontext`, `field`, `difficulty`, `competency`, `exercisetype` (for example `difficulty` can be `easy`, `medium`, or `difficult`). They are all encoded as children of the `extradata` element (itself a child of `metadata`) with their value in a single attribute (generally `value`).
- Pedagogical properties that are relations include `prerequisite`, `for`, and `is_part_of`. They are encoded within a `relation` child of the `extradata` element having `ref` elements as pointers.

Although the authors are likely to input the metadata annotations for the purposes of the ActiveMath learning environment, they are described and used with a semantic intent: using them implies these pedagogical properties by their meanings. This follows a similar approach as *the knowledge lifecycle* of Millard and Davis [MTD$^+$06] where the semantic nature of the annotations of learning objects are viewed as crucial for the long term value of the content.

The pedagogical annotations are used and displayed by several services of ActiveMath which we describe below: the learner-model, the tutorial component, the exercise system, the presentation pipeline, and the assembly tool.

2.3 Concepts in the ActiveMath Learning Environment

Learners use their web-browsers to access ActiveMath, directing it to the shared server assigned to their school or class. They log-in so that a personalized learning experience is offered. The functions of ActiveMath are, for most, accessed from the main-menu which is made, more or less of three main sections as illustrated in Figure 2.3:



Hello Sofia.
This is the main page of ActiveMath. Please choose one of the books.

English Books:

- Complete Content of LeAM_calculus
  Number of pages: 220
- Highschool Content of LeAM_calculus
  Number of pages: 166
- LeActiveMath Grade 11
  Number of pages: 40
- Secondary School Content of LeAM_calculus
  Number of pages: 160
- University Content of LeAM_calculus
  Number of pages: 192
- Zoology of exercises in LeAM_calculus

- derivative function
  Number of pages: 5
- model
  Number of pages: 4

Welcome to the evaluation server of the LeActiveMath research project.

To use LeActiveMath, you need:

- Firefox, at least version 1.5, or Internet Explorer 6 (other browsers may work, but are not our primary

Figure 2.3: A typical main menu of ActiveMath during the LeActiveMath project.

- the *pre-recorded books* offer predefined collections of learning materials which may include the launch of interactive activities. This is probably the first place to go when beginning the learning process.

- The user's *own books* are where the books created by the course-generation appear, and also where they can be modified by hand. This would be the best place to go to to be reminded of past learning experiences.

- The *tools* are accessible from the menu on the top. They include a variety of generic mathematical tools, a search tool, as well as tools to inspect the information the server has about the user.

The content, packed in books and sometimes accessed by the search engine, is where all the learning starts. Interactive exercises also start there, based on an invitation presented in the book which starts the exercise at a click.

The content items of ActiveMath are atomic entities which, typically, appear in pre-recorded and generated books. The same item can appear multiple times: in various books, in the search tool, and, for some, in the learner model inspector. This multiplicity stimulates the identification and rememberance, and therefore the learning, of the individual content items. We have called this approach *the book paradigm* of ActiveMath, a paradigm that has sometimes been criticized for its traditionalism and sometimes been recognized for its usability.

ActiveMath also supports learning by integrating tools within the learning experience. In the next section we first describe the core components that enable this experience, then the learning tools.

2.4 The Components of the ActiveMath Server

ActiveMath is a web-application exploiting the Java* Servlets specification. It runs as a web-server where server code is run in the Java Virtual Machine while code is also executed on the computer of the learners: displayed as HTML* or PDF*, run in JavaScript* or Java* applet*.

ActiveMath coordinates these components tightly inside the virtual machine and using web-service calls and HTTP* requests for the client. Each of the component has a specific role which we describe below.

Several overview papers about ActiveMath exist:

- the paper [MBG$^+$03] depicts the knowledge representation underlying the system,
- the paper [MGH$^+$06] sketches an overview of the components,
- the paper [MGLU09a] focusses on the semantic web dimensions behind ActiveMath,
- the paper [MGLU09b] highlights the adaptivity of ActiveMath to each culture.

2.4.1 The Core of ActiveMath

The web-server containing the ActiveMath web-application instructs its servlets to be initiated; this starts the web-application core which loads all the configurations, a sequence of properties* files in the `conf` directory. Among the configurations thus obtained are the various ports that need to be opened, the content collections to be loaded, and the notations to be incorporated. This configuration information allows services to be made available to other components of ActiveMath. The services are described by a Java* interface which specifies the methods and objects it can offer. The concrete implementation is decided by the configuration.

The core of ActiveMath permits the Model-View-Controller approach where the services are the access-points of the model, the controllers are made of small objects which interface between the two, and the view is provided by a set of Velocity* scripts describing the HTML* rendering of each of the web-pages.

2.4.2 The Content Storage

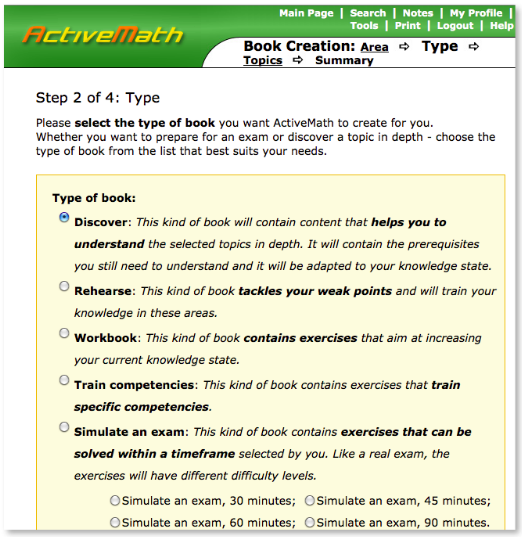
The entry point of the content into ActiveMath is an indexed storage which we describe in Chapter 4. The storage service offers a direct access to the individual content items of the collections of OMDoc files so that they are independent of the OMDoc file that they come from.

The content storage offers a simple set of methods to the other components, providing in-memory representations of the OMDoc fragments which can easily enter XML* processing as we describe below. It also delivers the references to and from content items – all within a reasonable performance for a classroom.

2.4.3 The Tutorial Component

The tutorial component of ActiveMath is a service and a set of web user-interfaces to suggest content items that the learner could learn with. For the task of generating a table-of-contents with recommended content items to reach particular learning objects, the tutorial uses the AI paradigm of hierarchical task planning (HTN).

The screenshot of Step 2 of the course-generator menu is on the right: at this point the student chooses the *type of book* she wishes to use.


ActiveMath
Main Page | Search | Notes | My Profile |
Tools | Print | Logout | Help
Book Creation: Area ⇨ Type ⇨
Topics ⇨ Summary

Step 2 of 4: Type

Please **select the type of book** you want ActiveMath to create for you.
Whether you want to prepare for an exam or discover a topic in depth - choose the type of book from the list that best suits your needs.

Type of book:

[✓ **Discover**: *This kind of book will contain content that **helps you to understand** the selected topics in depth. It will contain the prerequisites you still need to understand and it will be adapted to your knowledge state.*]

Rehearse: *This kind of book **tackles your weak points** and will train your knowledge in these areas.*

Workbook: *This kind of book **contains exercises** that aim at increasing your current knowledge state.*

Train competencies: *This kind of book contains exercises that **train specific competencies**.*

Simulate an exam: *This kind of book contains **exercises that can be solved within a timeframe** selected by you. Like a real exam, the exercises will have different difficulty levels.*

Simulate an exam, 30 minutes; Simulate an exam, 45 minutes;
Simulate an exam, 60 minutes; Simulate an exam, 90 minutes.

Other functions of the tutorial component also allow the suggestion of further content items (e.g., to deepen a particular topic).

The tutorial component uses two other components to perform its work:

- The learner-model, described below, is read in order to formulate the recommendation that is most appropriate to the learner's current competencies; for example, in a classical discovery scenario, it will attempt to include a mild introduction and exercises in growing difficulty for a concept that is deemed new to the learner With a learner that is known to have mastered said concept, it will attempt to offer more challenging exercises.

- The content storage is queried: the tutorial component queries the metadata elements and the relationships, so that the appropriate content items are chosen for the appropriate learning situation.

The tutorial component manipulates the content item using an abstraction of the content items structure and metadata called the Ontology of Instructional Objects (OIO) depicted in [Ull07]. This ontology abstracts away from the OMDoc annotation schemes used in ActiveMath which has allowed the tutorial component to offer its services to external systems, provided they could offer OIO-encoded knowledge about their content.

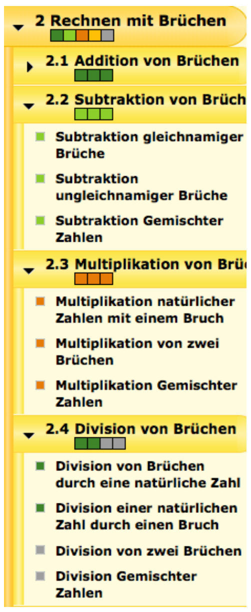
The HTN planner operators are expressed as rules which, given some conditions on the learner model, can apply if a set of instructional objects is met with a particular set of properties. The *mediator* is the sub-component responsible to convert the queries to retrievals from the content storage; similarly the mediator translates the metadata and relations delivered by the content storage to OIO objects. The mediator is described in [KUM06].

2.4.4 The Learner-Model

ActiveMath's learner-model is the component responsible to maintain a measure of what the learner knows or can do. To do this, it constructs a belief network in memory which gathers evidence and depth of the acquisition of a symbol, definition, theorem, axiom, or method. The evidences for each item are connected through conditional probabilities equations when there are prerequisite conditions or for relationships between the items. The learner model is explained in depth in [Fau07], based on a previous approach in [MvLB06] and with experiment-based refinements in [Doo10].

The learner model thus heavily depends on the authored content, in paticular the conceptual items and some of the relationships. To fully master this usage of the content, it is important for an author to possess a good understanding of the learner-model. However, little of the learner-model is displayed which makes it difficult to experiment with.

The most widespread display is in the *mastery bullets* pictured on the right: in some configurations, the table of contents' rendering, on the left, is enriched with a row of coloured dots indicating the average mastery depth of all conceptual content items of the page: grey means unknown, red, yellow, and green indicates low, medium, and high mastery.


2 Rechnen mit Brüchen
2.1 Addition von Brüchen
2.2 Subtraktion von Brüch
Subtraktion gleichnamiger Brüche
Subtraktion ungleichnamiger Brüche
Subtraktion Gemischter Zahlen
2.3 Multiplikation von Brü
Multiplikation natürlicher Zahlen mit einem Bruch
Multiplikation von zwei Brüchen
Multiplikation Gemischter Zahlen
2.4 Division von Brüchen
Division von Brüchen durch eine natürliche Zahl
Division einer natürlichen Zahl durch einen Bruch
Division von zwei Brüchen
Division Gemischter Zahlen

However, these indications have proved to be often confused with progress indicators within the linear path through a book and have been thus disabled by default.

Another method to show the learner model has been explored in [NBM07] but has not been kept.

The learner-model starts with zero evidence and is progressively updated by the events sent by the components. To date, only the exercise step events are considered, their metadata, and that of their enclosing content items are used to add to the evidence of an item (it can include competency acquisition and misconception diagnostic statements), triggering an upgrade or downgrade of the depth.

2.4.5 The Exercise System

The exercise system of ActiveMath loads the description of exercises stored in the OMDoc files and presents it to the learner. The exercise description can be either a graph of steps elements and connected by transitions between them, based on the user's input or a short mission describing the usage of an external *domain reasoner* which follows and guides the learner through the steps of a classical problem resolution.

Although the XML* elements necessary for this description are part of the OMDoc documents that an author may need to write, it is a particularly complex part because of the interconnected nature of the graph of interactions (implemented by the `interaction`, `feedback` and a few other elements). This has stimulated the creation of a separate authoring tool, shortly described in 3.3.5. For the structure of the elements, we refer to the thesis [Gog11].

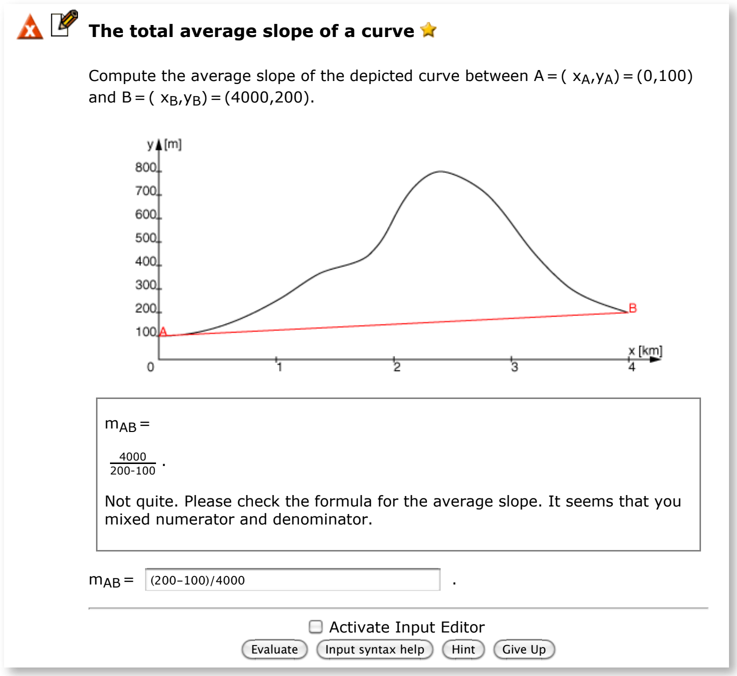
The exercise system interacts with the browser by a sequence of dialogs where an invitation or feedback is presented to the learner, then an input is received and analyzed (possibly using external systems such as a computer algebra system) and sent back. The exercise system can thus show to the learner her history of attempts and the computer's feedback. The verification of an exercise in the exercise system by a teacher or author involves exploring the possible ways a learner could be answering so as to proof that the system's answers.

A screenshot of the exercise system showing its evaluation capabilities is shown in Figure 2.4. We refer to [Gog11] and [Gog09] for more about the exercise system.

2.4.6 The Formula Input Facilities

Web-browsers and their associated standards do present features for users to input answers to questions in textual form as well as in multiple-choice forms; however there is no standardized input facility for mathematical formulæ. ActiveMath, justlike any mathematics learning environment, has components to allow the input of formulæ.

The most basic component is visible as a text-field in the user-interface of the browser: it is a simple text-input where learners can use one of a few linear syntaxes to input the mathematical symbols of the formulæ. The preferred syntax is close to that of the Maple computer algebra system. Although much appreciated for elementary answers (for example in the domain of fraction calculus),



The total average slope of a curve

Compute the average slope of the depicted curve between $A = (x_A, y_A) = (0,100)$ and $B = (x_B, y_B) = (4000,200)$.

$m_{AB} =$

$\frac{4000}{200-100}$.

Not quite. Please check the formula for the average slope. It seems that you mixed numerator and denominator.

$m_{AB} =$ (200–100)/4000 .

Figure 2.4: An interactive exercise in LeAM_calculus showing the evaluation capacities: it can recognize that the answer is, once computed, a well-known error, that of swapping y and x in the computation of the slope.

this input-field becomes quite hard to use with more complex expressions such as polynomials as reported by the evaluations (see Section 7.5). This method of input is supported in interactive exercises where parsing the linear input syntax is part of the exercise evaluation, as illustrated in Figure 2.4.

The most complete component for the input of mathematical formulæ is the Wiris Input Editor which is deployed as a Java* applet* in places where a user should input a formula: currently this includes the search tool as well as the computer algebra system and the exercise system (where it is optional).

The Wiris Input Editor is presented in [MEC$^+$06]. It offers the classical palette based input of mathematical symbols and has been designed to be easily customized and/or extended with new OpenMath* symbols by the usage of the *Wiris Domain Editor* availabile from jEditOQMath. The input editor can also load mathematical notations fom the ActiveMath notations (see next section). The input editor works, internally, on an annotated MathML*-presentation tree and routinely converts it to OpenMath and back, informed by the symbols' notations.

The evaluations have shown issues with this editor, in particular in its integration within the web-browser, but it remains as the de facto choice for a complete input solution. In Section 7.4 we propose approaches to ease the input of mathematical formulæ which have been implemented in the Wiris Input Editor.

The Wiris Input Editor submits its result as part of a form element which reads the OpenMath* from the applet* before submission, or as a separate URL where the content is stored on a regular basis on the server. A snapshot of the input-editor in use in the computer-algebra-system is shown in Figure 2.8.

Authors may, for a given installation, tune the input-editor's palettes, its symbols' notations and and their shortcuts. They should also make sure the input editor can be used to input the expected formulæ of the learners. Although it appeared reasonable, it took surprisingly long to realize that the input-editor was unable to enter simple things such as mixed fractions.

2.4.7 The Presentation Pipeline

ActiveMath is a web-server that is intended to present content to multiple learners in parallel, at least a classroom. The web-page which includes the links to functionalities as well as the content converted from the OMDoc language should appear almost instantaneously in the browser. However, many conversion tasks are needed for a web-page to be functional in the web-browser which can only display HTML* (with CSS*) or XHTML+MathML* (which we call the *presentation language*):

- The content fragments in OMDoc need to be converted to the presentation language, including all the formulæ in the appropriate language.

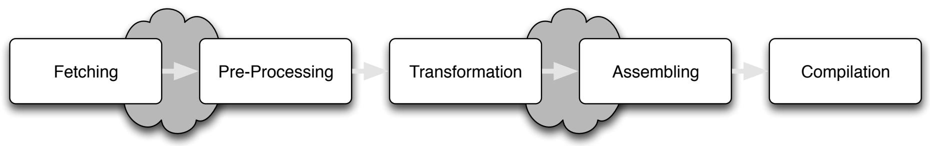


Figure 2.5: The presentation pipeline of ActiveMath: the content items are fetched from the content store on the far left and delivered within web-pages on the right.

- This conversion should be also usable in a context where OMDocs are dynamically generated, e.g., the exercise system.
- In a book, all rendered content items should display a witness of the content item's metadata, such as the title or the difficulty and offer, by a link, to provide more information about it (e.g., the bibliographical information).

There are more presentation languages than just HTML* and XHTML+MathML*. A user can request a printed copy, in which case a conversion to TeX* then PDF* should be done; similarly, in some ActiveMath versions, a user can request the content items to be read aloud: this conversion is done to text then to an audio media; in others, the presentation language is the scalabale vector graphics format of the W3C.

The conversion processes are thus assembled along a *presentation pipeline* which is sketched in Figure 2.5. It involves the content store (fetching), XML* manipulations (pre-processing), XSLT* transformations from OMDoc to fragments of the view language (transformation), merging in to one and annotating with the metadata (assembling), and potentially running a final conversion (compilation). The pipeline allows the conversion results to be cached so as not to be re-rendered at a later time (see the clouds in the Figure 2.5).

The transformation step is probably the most explicit conversion. It is a complex process described by an XSLT* sheet – a series of templates which instruct which presentation language is to be output by which OMDoc element. The XSLT* sheet is mostly written by hand but a part is dynamically generated: the set of templates to render mathematical formulæ.

This set of templates is, for the biggest part generated by the *notations* which are encoded in the OMDoc documents. Notations are written for each language. They are pairs:

- An OpenMath* element (a *prototype*) which is matched if the expression is found, the variables being replaced by arbitrary sub-terms.
- A MathML*-presentation element which describes the rendering of that same element, with variables replaced by the rendering of the corresponding term found in the prototype.

```
<symbolpresentation id="combinat1binomial_59_54" for="binomial">
   <notation precedence="1000" language="en">
     <math>
       <msubsup>
         <mo am:cross-ref="yes">C</mo>
         <mrow><mi am:precedence="0">n</mi></mrow>
         <mrow><mi am:precedence="0">m</mi></mrow>
        </msubsup>
     </math>
   </notation>
   <OMOBJ>
     <OMA>
       <OMS cd="combinat1" name="binomial" />
       <OMV name="n" />
       <OMV name="m" />
     </OMA>
   </OMOBJ>
 </symbolpresentation>
```

ru	fr	en	es
C_n^m	C_n^m	$\binom{n}{m}$	$\binom{n}{m}$

Figure 2.6: An example notation: the *big C* notation for the binomial coefficient in use in English-speaking texts. A series of such notations in each language allows one to render the same OpenMath expressions in multiple languages (on the right).

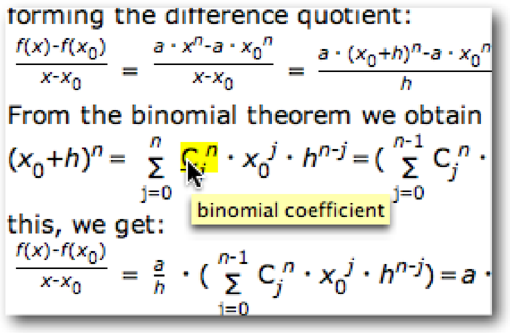
The presentation system is able to deliver a *value-added formula rendering* illustrated in Figure 2.7: the title of symbols are provided as tooltips, and a click lets the user open a menu to see, for example, definitions of this symbol. See 7.4.

An example notation and its usage in the rendering of a document is in Figure 2.6. We refer to [MLUM06], [Man05], and [Lib07] for more information about the notation system.

The notations are expected to be input by authors who want to extend the set of symbols of ActiveMath – for example, adding a new concept or a mathematical symbol which differs somehow to previous existing symbols. This requires a good understanding of OpenMath* and of MathML*-presentation. At time of loading the content, the system behind the presentation pipeline collects the notation elements and transforms them to XSLT* templates. These templates are then made part of the presentation pipeline.

2.5 Learner Tools

The ActiveMath learning environment aims at a pedagogy that is close to moderate constructivism as explained in [MMU$^+$07]. Within this pedagogy, the guidance to learning can be provided by learning materials, but the usage of tools to support the explorations of the learners are quite important. The tools are instruments that the learner can use to manipulate the mathematical knowledge and content: for example perform computations, or find and organize learning content. The usage


forming the difference quotient:

$$\frac{f(x)-f(x_0)}{x-x_0} = \frac{a \cdot x^n - a \cdot x_0^n}{x-x_0} = \frac{a \cdot (x_0+h)^n - a \cdot x_0^n}{h}$$

From the binomial theorem we obtain

$$(x_0+h)^n = \sum_{j=0}^{n} C_j^n \cdot x_0^j \cdot h^{n-j} = (\sum_{j=0}^{n-1} C_j^n \cdot$$

binomial coefficient

this, we get:

$$\frac{f(x)-f(x_0)}{x-x_0} = \frac{a}{h} \cdot (\sum_{j=0}^{n-1} C_j^n \cdot x_0^j \cdot h^{n-j}) = a \cdot$$

Figure 2.7: Value-added presentation of formulæ in HTML+CSS: the tooltip is shown on the big C to indicate its meaning, the yellow rectangle encompasses the C and its arguments, n and j: the smallest sub-term containing the big C.

of tools is complementary to the reading and exercising activities – they transform them into a more self-guided process, hence making them more effective.

In this section we detail the tools that ActiveMath offers out of the box and how they are relevant to the author. Multiple other tools have been created for the specific purposes of content projects but we shall not describe them.

2.5.1 The Search Tool

The search tool of ActiveMath aims at exploring the pool of learning content made available. It offers both a simple search, which can be used by *just typing* and an advanced search where precise queries can be formulated about the content items.

The search tool of ActiveMath can support the learner in finding content items she has already read or new content items that may help her to deepen a particular learning direction. The search tool is described in depth in Chapter 5.

The search tool, together with the course generation function, is an important tool for authors to use when checking content. Contrary to the careful selection of content items one can perform by creating a table-of-contents and recommended it to learners, the search and the course-generation will crawl through the whole content storage. It will be common, for example, that learners encounter mild duplicates when searching. In some cases this can be disorienting or inconsistent in language. Depending on the learning process, this can be a real problem and should be guarded against by the author; in other cases, this can be viewed as an advantage for learners to open their perspectives to other ways of doing mathematics.

2.5.2 The Assembly tool

We have described in Section 2.2.1 how ActiveMath organizes content items in books by their table of contents. In Section 2.4.3 we have described how the course generator can suggest books by producing a table-of-contents.

The assembly tool is a user-interface to edit the table-of-contents. It allows users to modify the table-of-contents displayed in the section *my books* of the main menu by rearranging sections, pages, and the list of items of each page, and by renaming them. Content can be added to the assembly tool by drag-and-dropping a chapter in the table-of-contents, or by drag-and-dropping an item link, e.g., in the title of an item in a search result.

The assembly tool is a Java* application started by a JNLP* file which carries the authentication of the user. It communicates to the other components, the user-book storage and the content-storage, by XML-RPC*.

A picture of the assembly tool is in the Figure 2.2. More information about it is available in [Hom06].

2.5.3 The Concept Mapping Tool

Exercising the connections between content elements is another knowledge organization task which is recognized to help the learner manipulate, inspect, and proof what she knows of the content. This is the objective of the concept mapping tool, a user-interface that lets users edit graphs of content items with connections between them and check these against the relations stored in the content.

The concept mapping tool is connected in a similar way as the assembly tool: started with JNLP* and uses XML-RPC* services. It can be started on the simple initiative of the learner, to sketch relationships she understands, or as an exercise to rehearse the connections between the content items that have just been learned about. We refer to [MKH05] for details about the concept-mapping tool.

2.5.4 The Computer Algebra System Tool

In Section 2.4.5, we have indicated that the evaluation of the learners' answers can use a computer algebra system. A very simple application of this is displayed in the form of the *computer algebra system* tool: it is a simple dialog where the students input a mathematical formula and the computer algebra system *computes* this formula.

Although this approach does not allow a particularly fine control of the computation (e.g. there is no way to request an evaluation in floating point, or to define variables to be used later), this tool can be put to good use for computational duties where the learner *wonders* about a given computation, or simply does not want to take the time to compute herself.

Technically, the computer algebra system tool is a special interactive exercise and shares the same formula input. This implies that such copy-and-paste facilities as those described in 7.4 also apply here: a learner can thus transfer a formula from the presented content, adjust it, then compute with it. A picture of the computer algebra system tool at work is shown in Figure 2.8.

2.5.5 The Function Plotter

The final tool integrated in the ActiveMath tool set is the function plotter which can display the graph of a function.

The function plotter in ActiveMath is derived from the Java Components for Mathematics of David Eck and others at `http://math.hws.edu/javamath/`; it is run as a standalone applet*.

The function plotter has a simple communication link to the host ActiveMath: it can receive *clip-URLs* which are drag-and-dropped by users that click the context menu on a term of formula in ActiveMath. The plotter then fetches the OpenMath* content and transforms it into the proprietary linear syntax of the plotter. A screenshot of the plotter receiving drops from another source of function is shown in Figure 2.8.

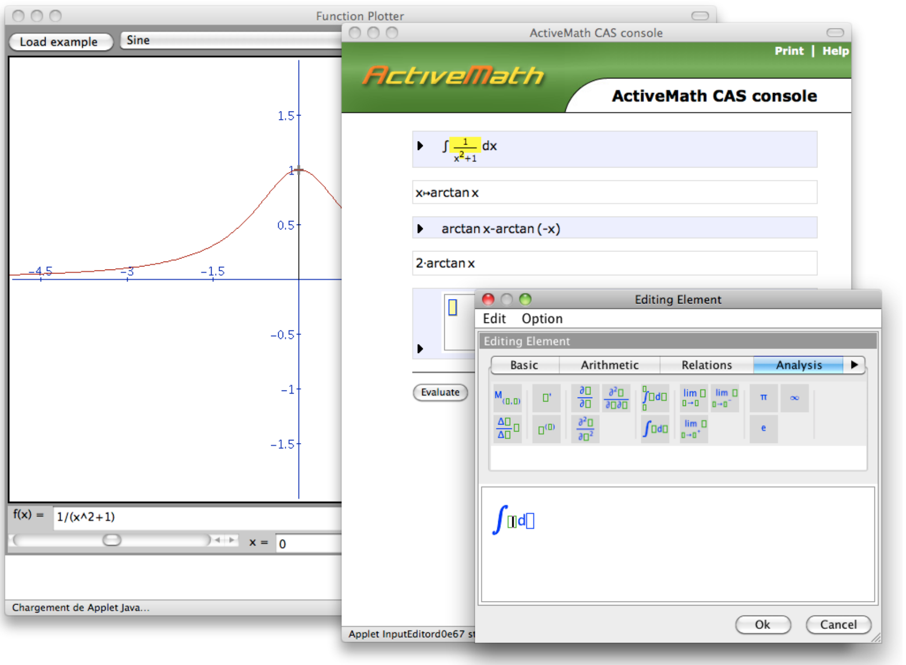
The plotter is a tool that is almost independent of the author's content. However authors should still verify that the drag-and-drop still works for terms they consider useful to be plotted.

2.6 Conclusions

In this chapter, we have provided an overview of the ActiveMath learning environment, which shows its broad spectrum, far beyond the simple presentation of static content on the web.

This broad spectrum is both a chance and a challenge to the authors who create content to be used in ActiveMath: it opens unprecedented possibilities for the learners to take an active part in their learning, but it also poses an unprecedented challenge to the orchestra leader that is the teacher and, often, the author: similarly to the instrumental orchestration approach [DT08], the teacher and content author must use the right words at the right time to describe what a learner could or should do. This challenge is multidimensional:

- In mathematical terms, e.g., when answering such a question as *how to express the resolution steps to this problem?*.
- In technical terms, e.g., when answering *how to express the steps to use on the computer to discover this graph view?*.



$\int \frac{1}{x^2+1} dx$

$x \mapsto \arctan x$

$\arctan x - \arctan(-x)$

$2 \cdot \arctan x$

f(x) = 1/(x^2+1)

x = 0

Figure 2.8: The function plotter of ActiveMath where the student requests the graph of the plot of each of the functions she obtains by the computer algebra system.

- In pedagogical terms, e.g., when answering *how much theory should we provide before an exercise is suggested?*.

ActiveMath's originality lies in the integrated approach to this orchestration based on semantics foundations. From the point of view of a teacher, ActiveMath strikes by the freedom it offers to the learners: *Based on the system's competence assessment, a learner is presented with theoretical and interactive material and has great freedom in the way s/he learns.* [TBN09]. This study [TBN09] depicts the great diversity of the possible interactions a learning system could offer. This highlights how detailed the knowledge of the system should be for anyone that suggests the usage of ActiveMath for learning, such as a teacher or author.

In the remainder of this thesis we describe authoring-relevant facets of my contribution to the ActiveMath learning environment.

The ActiveMath project and its research is ongoing. We refer to `http://www.activemath.org/` for newer references.

Part II

Authoring for ActiveMath

Chapter 3

WYCIWYG: Authoring with jEditOQMath

Having described the content model to be used in the ActiveMath environment (Chapter 2), I will now discuss to the authoring tool I propose to use: jEdit-OQMath.

This tool has strong roots in the development history of the OMDoc language and the ActiveMath environment: both evolved during the profiling of an authoring practice and ActiveMath was created as a server framework that can exploit author-produced OMDoc contents. I first describe the practice it emerged from. Among others, jEditOQMath followed the permanent care for a readability of OMDoc fragments in early authored bits and documentations (until currently in the OMDoc book [Koh06]): this chapter describes the workings of this practice while Chapter 9 describes experimental results. I then present my own contribution: an editing practice that focusses on giving to the user (the *author* mentioned in Chapter 1) access to view the ***effects*** of their actions. An argument is presented to describe this practice as a form of *direct manipulation*.

This chapter has evolved from [LG06] which was the first presentation of the jEdit-OQMath authoring tool; a concentrated version of it has appeared as [Lib10].

3.1 Mission

The user-interface described in this chapter is intended to enable an easy input, review, and modification of content for the ActiveMath learning platform described in Chapter 2. In particular, the formulæ aim to be semantically processable.

The authors' work is the production of content whose objective is to be an essential ingredient of the experience of learners. The result of authoring only makes sense in the possible usages of it. Among these usages, the main one is the direct learning experience he can see in the learning environment.

3.2 Set-up for jEditOQMath

I propose that authors use the three following tools for authoring. The usage of these tools is summarized here and described with all details in the tutorials described in Chapter 9:

- an **Authoring ActiveMath Server**: a server running on the author's machine that reads the files stored on the local computer
- a classical **desktop file-management** interface to manage files, grouped in content-collections (see Chapter 4); files are *content sources*.
- a **text-editor** to modify the content of the files

All three tools have to be used in conjunction to obtain the results of authoring, i.e. achieve the readiness of the desired experience of the learning environment. Once satisfied, the content collections are transferred in order to become published, e.g. on a server or on a repository for others to use.

3.3 Development History

In this section, I present the development history of the ActiveMath authoring practice. I hope to present in such a way that the reader better understands the evolution of the software.

3.3.1 Hand edited XML

As explained in Section 2.2, the OMDoc language was born with the double purpose of being a way to communicate via automated theorem provers as well as acting as a container for user-orientated texts: A strong focus existed on the expected semantic-based interoperability. Although this fact is purely historical it had a deep impact both on the development of ActiveMath, OMDoc and even the authoring practice described here. Traces of this evolution can be seen in early papers about OMDoc and ActiveMath (e.g. [Koh00]), [MBG$^+$03] or [gJSBF$^+$00]).

During this period, the model of OMDoc content has evolved as a carefully designed XML* language for its ease of readability and input; based on a highly engineered DTD*, OMDoc documents and fragments of them were created, exchanged, discussed, exploited, rendered, and re-edited. Thanks to the DTD*, the fragments contained even sections of standardized XML* with a specification coming from outside (most notably Dublin-Core* and OpenMath*); as a result, their readability was maintained.

This is the period which saw the birth of the authoring practice of ActiveMath content that I propose here. Moreover, during this period, the evolution of the content model took place: the OMDoc model was refined as more content was encoded. The metadata model also benefitted from refinement as its exploitation grew. (E.g. the introduction of competency annotations for use in the course generation – these are described in Section 2.2.2.)

3.3.2 QMath

During that period, several tools appeared to generate the OMDoc content from other formats. In particular the TeX language appeared to be appropriate for this purpose (see, for example, [Koh04, BM06]); moreover, attempts from styled documents also appeared [GP03]. Another tool, QMath, offered the approach of a compact syntax to generate the OMDoc documents: I have proposed to adopt it and have stimulated its development. QMath is described in [Pal06]. This tool became the core of the recommended practice described within an *authoring kit* which packaged installation and authoring documentations.

The model of the OMDoc documents for ActiveMath kept evolving. The author of QMath, Alberto Gonzáles Palomo, gracefully updated QMath but this loop became extremely dense with some properties not being possible to input early enough. Moreover, the editing features for QMath sources were limited to syntax coloring and line-number-based error-reporting. The biggest issue, which is still current as of this writing, was the impossibility of locating in the QMath source the origin of an error found in the OMDoc output.

3.3.3 OQMath

At this point, OQMath was created, with its existence first formally announced in August 2003. This is the central encoding contribution of this thesis. OQMath's creation was stimulated by the possibility of a more supported edition (that of XML* documents) and of a more supported validation (reference errors that can be reported with a line number).

The OQMath tool is a processor that takes documents and extracts the notation definitions and the formulæ islands, gives them to QMath, and replaces the formulæ islands in the output. This allowed authors to edit OQMath files as valid XML* files, supported by the guidance of DTD*, and obtain validation on the authored for both – the simple encoding (formula, DTD) and the larger reference errors. Moreover, OQMath settled the issue of changing schemas satisfactorily.

In one area of content, OQMath showed itself hard to manage: the interactive exercises whose structure, described in [Gog09], ressembles that of a complex graph with many *go-to* interspersed. A tool called ExaMat that is dedicated to interactive exercise authoring is described below.

3.3.4 jEditOQMath

For the OQMath files to be edited, an editor has been chosen: jEdit. The reasons for this choice are multi-faceted: the editor is open-source, written in Java* (which I knew), is widespread and implements the classical gestures (as opposed to, say, Emacs or vi for which all normal keyboard shortcuts have to be re-invented), and it offers a reasonable XML* support.

I have been exploring several other XML* editors: visual editors presented the document in a way that actually takes more visual space than a careful OMDoc source and did not simplify the appearance hence were discredited.[1] The last candidates remaining were the Morphon XML* editor[2] and some several source editors. The first is now discontinued but was the only editor that allows a sequence of text interleaved with a link to be presented in a single line; its commercial nature and discontinuation made us abandon it.

The second, the category of source editors, was rich and still is: the Emacs nsgml mode came out on top amongts the editors that worked on the content but was clearly rejected as non-intuitive for newcomers because of its highly non-standard basic gestures. jEdit remained as an editor that is easy to tune, that is feature-rich for the XML* documents, and that is intuitive in its gestures from its first contact; its open-source development model and the Java* technological basis both were compatible with the rest of the ActiveMath software traditions.

3.3.5 ExaMat

Stimulated by the difficulty to keep an overview of the XML* documents or interactive exercises, as well as the need to accomplish a tool that is easy for beginners, the ExaMat authoring tool has been developed, [GT07].[3]. The ExaMat tool is a browser-based editor which allows easy input of exercise fragments, their interaction dialogs, and their transitions; it has been developed along with the exercise system's evolution. However, occasional incompatibilities appeared where authoring particular features of the new exercise system was found to be impossible with it. ExaMat does not manipulate files that authors can arrange, nor does it support versioning; authors can use ExaMat and export the files so that they can be incorporated to a collection and, later, be part of a recorded book or be used by the tutorial component – but this step is not part of a revision cycle as we describe below. ExaMat is tightly bound to the exercise-system of ActiveMath so that it allows a fast preview to try the exercises but this preview does not happen within the overall ActiveMath experience. It has often been observed that

[1]Among such approaches the semantic wiki SWIM has this approach: it allowed to edit *generic XML* by presenting elements as tables, hence nested elements as nested tables, see [Lan07].

[2]The Morphon XML editor was a Java*-based editor for generic XML customized mostly by CSS*, see `http://xml.coverpages.org/morphonAnn.html`.

[3]ExaMat is the *politically correct* naming of this tool whose original name, the politically incorrect *eXtasy*, remains in the mouth of all developers and most users.

errors only appear when the exercises are loaded into ActiveMath. Last, but not least, ExaMat's most common practice creates identifiers in the form of long half-random numbers which are, I believe, contrary to the needs of long-term content development and the basic readability whereby an author recognizes elements of his content.

3.4 Tame XML Source Authoring with jEditOQMath

The activity I propose to authors follows the classical text-source-authoring practices that I have described for TeX and MediaWiki in Chapter 1: its input is provided by the editing of a text document and its exploitation is done in a separate *preview* space.

The source editing I propose, however, is not *free plain text*, which is far too liberal and cannot provide any useful information to a system to deliver rich services. Similarly to most other source language, it is structured along a basic grammar. Moreover, many of its ingredients only make sense when delivered by the ActiveMath learning environment.

Interactions in this model are done at three levels, each depending on the previous: interactions with the files, interactions with the (textual) source, interactions with the XML*, OpenMath*, and OMDoc model, interactions with the ActiveMath exploitation.

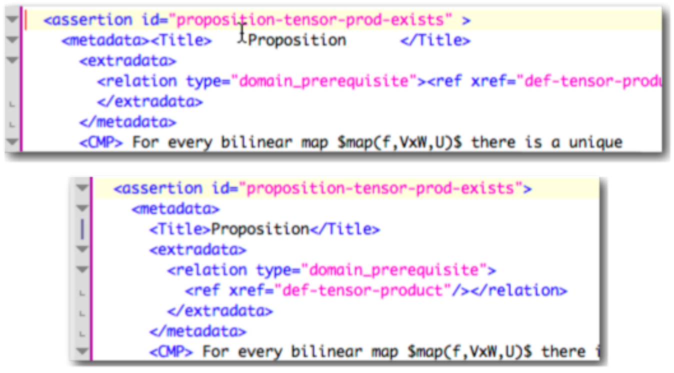
The author interacts with representations of the content at each of these levels and uses commands to go from one to the other and back.

3.4.1 Interaction with Files

As is the case with most data collection, the materialization of the content of a project is made in files that live in the desktop file-systems of the author. This allows multiple preservation, archiving, and sharing mechanisms that are available for them: send by mail, download, synchronize over subversion, put aside, backup: those are all activities that can be exercised with files.

As I shall explain in Chapter 4, ActiveMath stores the content files within directories called *content collections*: these form the unit of exchange. The exchange of these directions is at the heart of two important activities:

- re-use of existing content materials is best done by re-using full content-collections, as explained in Chapter 6: this is done by exchanging the files of the content-collections using such tools as a web-download, an FTP* transfer, or a versioning server



```
<assertion id="proposition-tensor-prod-exists" >
  <metadata><Title>    Proposition      </Title>
    <extradata>
      <relation type="domain_prerequisite"><ref xref="def-tensor-prod
      </extradata>
    </metadata>
    <CMP> For every bilinear map $map(f,VxW,U)$ there is a unique
```

```
<assertion id="proposition-tensor-prod-exists">
  <metadata>
    <Title>Proposition</Title>
    <extradata>
      <relation type="domain_prerequisite">
        <ref xref="def-tensor-product"/></relation>
      </extradata>
    </metadata>
    <CMP> For every bilinear map $map(f,VxW,U)$ there
```

Figure 3.1: An horizontal and a vertical indenting style

- deployment of the content-collection into a learner-oriented platform. For example, the installation into a school server is also done by the exchange of content-collections.

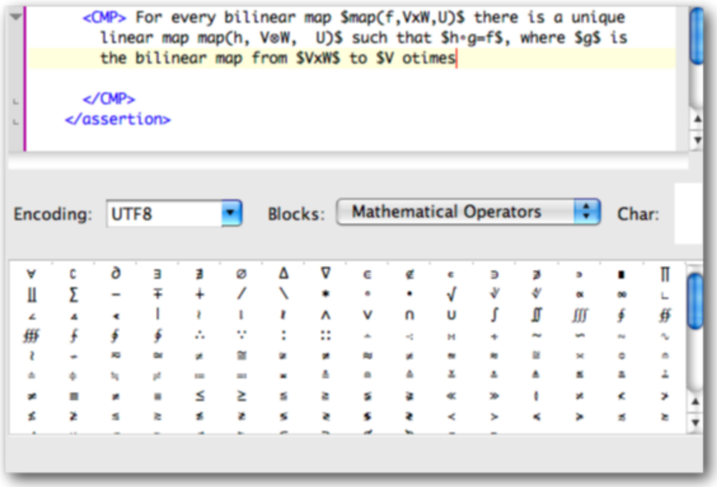
Files are also the starting point for the start and stop of the ActiveMath server: in the standard authoring practice, each author has his own authoring ActiveMath. This allows him to sculpt the learning environment to fit his needs in fine detail before he can reproduce his setup, e.g. on a school server. This requirement has shown to be a challenge in terms of memory and processing capability for the authoring machines at early times of the authoring platform.

Last but not least, an author can *open* a file – e.g. simply double-clicking it: depending on its type, this will start the editing process of this file. For OQ-Math files, the format for the files of textual content in jEditOQMath, this editor is opened and the text is presented. Picture files are opened in a picture editing environment, such as Adobe Illustrator (see Chapter 1).

3.4.2 Interactions with the Source

In this section, I concentrate on editing the textual source files. These constitute the core of the authoring activities but need to be supplemented by several other file-types, including pictures, animations, and the content-descriptor. The textual source files format is called **OQMath**. OQMath files are XML* files that are edited as plain text files. They are kept analyzable thanks to the grammar they follow which is ensured by a DTD*. They are kept readable by simple arrangements of text edition. Finally, they are compact thanks to multiple hidden values provided by the DTD.

The manipulation of the text source follows the classical text-editing practices: copy-and-paste, simple keyboard input, drag-and-drop, search-and-replace... All



```
   <CMP> For every bilinear map $map(f,VxW,U)$ there is a unique
     linear map map(h, V⊗W,  U)$ such that $h∘g=f$, where $g$ is
     the bilinear map from $VxW$ to $V otimes

   </CMP>
  </assertion>
```

Encoding: UTF8   Blocks: Mathematical Operators   Char:

Figure 3.2: Inputting a tensor product with the character table and with the abbreviations

of these gestures are widely known and implemented in an intuitive fashion by the classical editor, jEdit. Characteristic for a user-friendly input syntax is the fact that the XML* nature of the OQMath files is sufficiently flexible to provide sufficient freedom to personal desires in the placement of tags. Two ways to place the tags are depicted in the two indenting styles of Figure 3.1.

Character Tools

One of the fundamental characteristics of the textual sources is the link between the text and the keyboard keys to input it. The mathematical language has a tradition of being expressed on blackboards and being very compact. Mathematicians do not find enough characters on a normal keyboard to input it as it uses a large amount of extra symbols. While the TeX language (see Chapter 1) has addressed this by the usage of macros in the sources, written with words such as `\Delta` to represent the capital greek letter D, the OQMath format relies on the Unicode* standard which standardizes most such symbols. Provided the editor lives in an environment which can display these characters (common nowadays), the source can be made much more readable than TeX: a greek letter capital Delta is viewed as Δ and not as a long macro and is, thus, much more similar to a notation on the blackboard; similarly a tensor-product is written $\otimes$ instead of `\otimes`. Input of such characters can, however, be a challenge - jEditOQMath proposes two solutions for this which are both depicted in Figure 3.2:

- the character map is a pane at the bottom of the editor. It shows all the Unicode* ranges and, when available, the range of characters. This provides a basic means to access a character but requires the table to stay open

- an alternative method is provided by jEdit's abbreviation command: it is predefined with all TeX macros whose list I have been taking from the entity definition of MathML* – for example the typing D-e-l-t-a then CTRL-semicolon (or CMD-semicolon) converts this sequence of characters to Δ.

Populating the Source

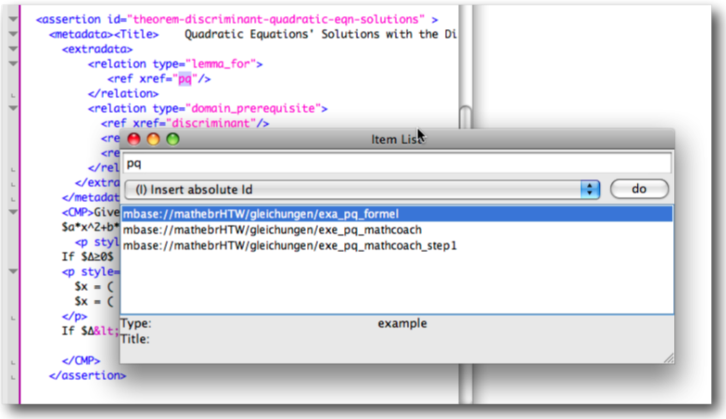
Since interacting with the source text is among the most important authoring activities and it offers considerable freedom, facilities are provided to help with *inputting the right text*. In the next section, I shall see how to *make sure the text is correct*.

First-time users of jEditOQMath will start by using the templates, to follow the *default practice*. The templates create the skeleton of a new document, all common-use content-items (definitions, examples, theorems...), and commonly input constructs inside the items (formulæ, images, ...). Because templates are made to create content that is modified thereafter, the expected places to be filled are marked with template-zones, islands starting with « and ending with ». A first run through a template is done by jumping from template zone to template zone, by pressing a button.

Transferring[4] from outside of jEditOQMath is also a way to start inputting content. An area where input is error-prone is the **input of references**: they are of utmost importance to construct the items' net that contributes to the intelligent behavior of ActiveMath as well as to support navigation. References can be dragged-and-dropped from an ActiveMath web-page (e.g. the title of an item in a book-page in ActiveMath, see Chapter 2): Another way to insert references is to use a search-box called the *searchable items list* – a small pop-up that allows to find items by IDs in order to navigate to them or to reference them. This is depicted in Figure 3.3. The following places and elements make sense and are used by the drag-and-drop; the software tries to insert the shortest reference possible so as to keep readability:

- within `CMP` elements, `textref` elements representing hyperlinks are inserted
- within the `relation` element of the `extradata` elements, only `ref` elements make sense
- in other places, it is not known... so the reference itself is dropped

The reference input gestures of jEditOQMath are similar to many other web-based tools in their input of a reference (e.g. ExaMat*, MOT*, Confluence*) but instead of clicking through dialogues for the right choice, the drag-and-drop paradigm is



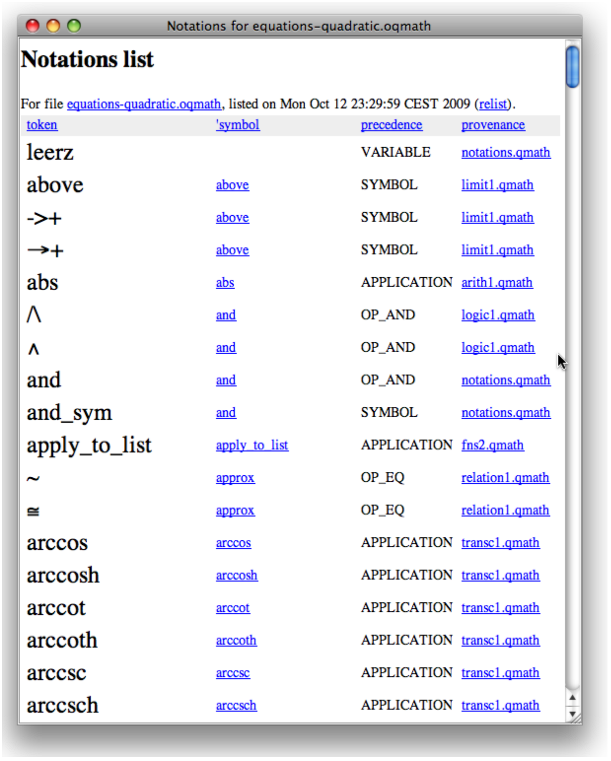
```
<assertion id="theorem-discriminant-quadratic-eqn-solutions" >
  <metadata><Title>    Quadratic Equations' Solutions with the Di
    <extradata>
        <relation type="lemma_for">
          <ref xref="pq"/>
        </relation>
        <relation type="domain_prerequisite">
          <ref xref="discriminant"/>
          <re
          <re
        </rel
      </extra
    </metadat
    <CMP>Give
    $a*x^2+b*
      <p styl
    If $Δ≥0$
    <p style=
      $x = (
      $x = (
    </p>
    If $Δ&lt;

    </CMP>
  </assertion>
```

```
mbase://mathebrHTW/gleichungen/exa_pq_formel
mbase://mathebrHTW/gleichungen/exe_pq_mathcoach
mbase://mathebrHTW/gleichungen/exe_pq_mathcoach_step1
```

Type: example
Title:

Figure 3.3: The searchable item-list invoked on the text *pq* suggesting a few items and, in this case, to insert a reference.



Notations list

For file equations-quadratic.oqmath, listed on Mon Oct 12 23:29:59 CEST 2009 (relist).

token	'symbol	precedence	provenance
leerz		VARIABLE	notations.qmath
above	above	SYMBOL	limit1.qmath
->+	above	SYMBOL	limit1.qmath
→+	above	SYMBOL	limit1.qmath
abs	abs	APPLICATION	arith1.qmath
/\	and	OP_AND	logic1.qmath
∧	and	OP_AND	logic1.qmath
and	and	OP_AND	notations.qmath
and_sym	and	SYMBOL	notations.qmath
apply_to_list	apply_to_list	APPLICATION	fns2.qmath
~	approx	OP_EQ	relation1.qmath
≅	approx	OP_EQ	relation1.qmath
arccos	arccos	APPLICATION	transc1.qmath
arccosh	arccosh	APPLICATION	transc1.qmath
arccot	arccot	APPLICATION	transc1.qmath
arccoth	arccoth	APPLICATION	transc1.qmath
arccsc	arccsc	APPLICATION	transc1.qmath
arccsch	arccsch	APPLICATION	transc1.qmath

Figure 3.4: The notations list presenting all notations' definitions within the current document.

used. A complementary paradigm could be implemented where each reference is auto-completed and is a (latent) hyperlink.

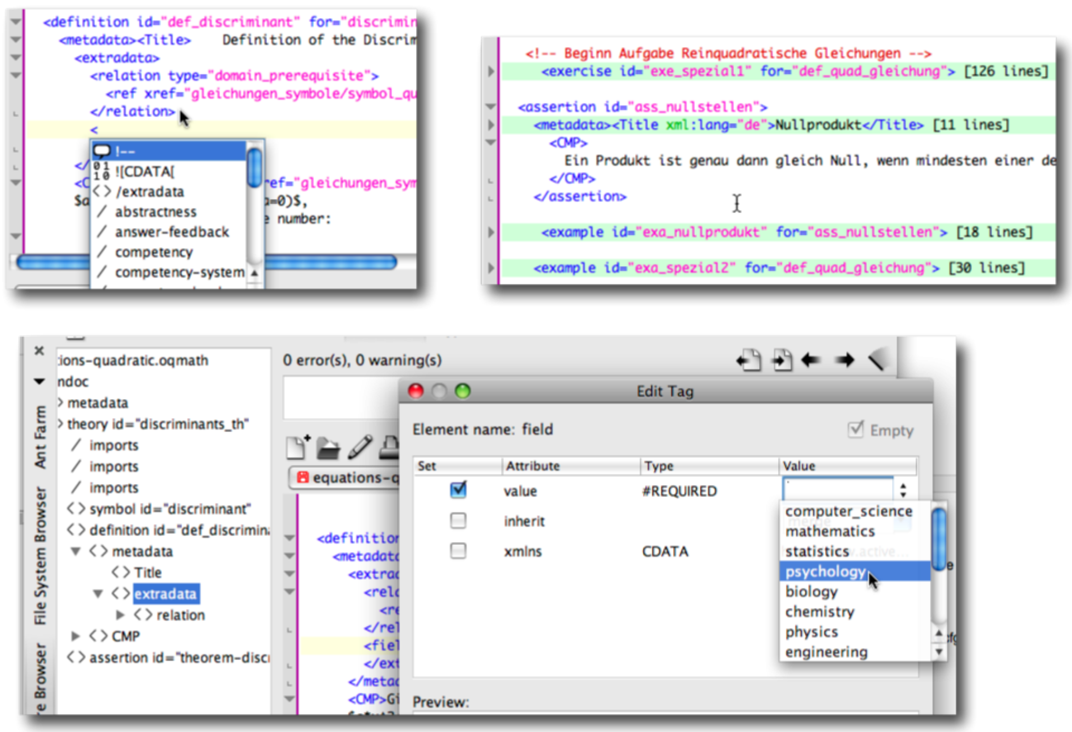
Copy-and-paste of formulæ is a further area where a transfer from outside is offered. We shall see in Chapter 7 the efforts to enable the transfer of formulæ from multiple places into jEditOQMath. This paste function is of use to discover the encoding of mathematical formulæ from the broad web but can only work for formulæ that “make sense" in the current context: they are converted to OpenMath* using the symbols available and to QMath using the available notations.

As I have described in Chapter 2, formulæ in OMDoc are to be encoded semantically with the OpenMath* standard. The OQMath format simplifies this task by allowing the **mathematical formulæ** to be **input in short syntax** based on notation definitions that authors can extend. These notation definitions are the main input for the QMath [Pal06] tool which transforms formulæ written in the short syntax into OpenMath* XML* objects during the OQMath process. Support for editing these formulæ is provided by a browsable list of the defined notations which is depicted Figure 3.4, by the copy-and-paste of formulæ described in Chapter 7, and by an immediate trial tool, the *QMath experimenter*. All these features take the notation definitions of the OQMath file being edited. This may pose the problem of missing notations but has the advantage of flexibility of formulæ input, a freedom which I have seen many mathematicians express preference for and which might be one of the key success of the TeX programme.

The **input of the XML tags** that constitute the structure of the document is supported by the grammar of the collection of the document, which is created at collection initialization. It has been a deliberate choice to keep the OQMath file in XML* format: the number of tools supporting its edition is important and this support is fundamental for users that are unsure of the possible inputs. Below I shall present a breadth of support offered within the paradigm of *validating* the file. Before validating, jEdit uses its XML plugin to offer a pop-up indicating all possible children, at the input of the start-tag character < (according to the DTD*). Moreover, a tag can be *edited* which displays a window documenting each attribute and its allowed values. Finally, jEdit allows the user to hide parts of the XML tree, by *folding* them with a click on the small triangle on the left of the tag in the source display. This has been most often used by authors to maintain a good readability. All these features are visible in Figure 3.5.

Finally, because the source is made of XML* syntax I have worked on **refactoring methods**. Currently the refactorings include the initial addition of the XML* elements for a translation, the clean-up of the XML file (to make it more readable), and two suggestion services: suggest imports to propose imported theories and preview metadata inheritance (see Chapter 4). All of these methods are based on three complete steps (XML parsing, modifications to the in-memory XML-tree, and replacement by the output of the revised documents), hence are generalizable.

[4]As described in Chapter 7 as a generic term for user-interface actions such as copy-and-paste and drag-and-drop.



```
<definition id="def_discriminant" for="discrimin
  <metadata><Title>     Definition of the Discri
    <extradata>
      <relation type="domain_prerequisite">
        <ref xref="gleichungen_symbole/symbol_qu
      </relation>
      <
```

```
<!-- Beginn Aufgabe Reinquadratische Gleichungen -->
  <exercise id="exe_spezial1" for="def_quad_gleichung"> [126 lines]

<assertion id="ass_nullstellen">
  <metadata><Title xml:lang="de">Nullprodukt</Title> [11 lines]
    <CMP>
      Ein Produkt ist genau dann gleich Null, wenn mindesten einer de
    </CMP>
  </assertion>

  <example id="exa_nullprodukt" for="ass_nullstellen"> [18 lines]

  <example id="exa_spezial2" for="def_quad_gleichung"> [30 lines]
```

Element name: field

Set	Attribute	Type	Value
✓	value	#REQUIRED	
☐	inherit		
☐	xmlns	CDATA	

Figure 3.5: A few of the XML-editing features available in jEditOQMath: pop-up indicating the allowed children after the input of the less-than sign (top-left), collapsing to keep only all details needed visible (top-right), structure browser and form-based editing of an element's attributes (bottom).

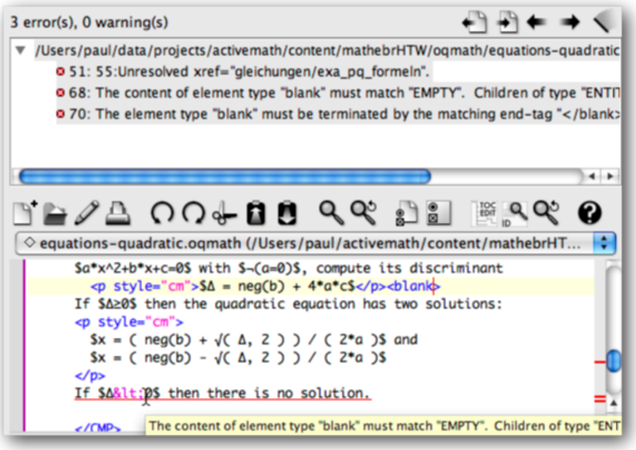
I have explained in this section all the facilities to input and maintain a readable source; the result of this editing process is a file that should conform to particular standards, the application of which is described in the next section. It will enter the content of the learning platform ActiveMath which the author can then check for its quality.

3.4.3 Exploitation of the Content: the Validation and the Build

Having input the necessary text, the author wishes to making sure of the results. The source is not the goal of authoring, it is only a means towards the creation of an anticipated learning experience. This fundamental expectation implies that the author has **to make sure the content is understood** and **enables the intended learning experience**.

The first aspect, make sure the author is understood, is done by the validation of the sources. Validation consists of four processes. Two of these processes are part of the build triggered by the users in order to transmit the authored content to the learning environment:

- well-formedness is checked every time the user saves the file and it is displayed with a text on the bottom of the window, and more importantly in the form of the collapsing-triangles as seen in Figure 3.1. This positive sign is fundamental: without it, no parsing is possible hence nothing else can be analyzed. The most common such issue is that of a wrong closing-tag. An error is displayed in the error list; the user can go to the source position where they can repair this. Well-formedness issues are generally trivial to fix.
- XML* validation is analyzed at every file-save as well; it evaluates that the XML structure matches the grammar specification provided by the DTD*. Again, errors are reported in the error-list, thus enabling an easy fix. However, most of these errors are non-fatal, and a (maybe partial) further usage of the source remains possible.
  One common occurrence of validation errors is at DTD-upgrade time: this process follows an update of the DTDs attached to the OQMath files, typically following evolutions of the ActiveMath server (during the years of this PhD, changes in the allowed metadata* annotations were the most frequent ones): in this process, each source file needs to be opened,validated, and adapted according to the new model, for example possible educational levels which get refined need adjustments for them to match the educational levels that learners will be choosing in an updated ActiveMath.
- QMath, when converting islands between dollar-signs to OpenMath*, may report errors indicating a trouble at procession notations or formulæ. The OQMath process makes sure that the error is reported on the line of the formula.



```
3 error(s), 0 warning(s)
▼ /Users/paul/data/projects/activemath/content/mathebrHTW/oqmath/equations-quadratic
    51: 55:Unresolved xref="gleichungen/exa_pq_formeln".
    68: The content of element type "blank" must match "EMPTY".  Children of type "ENTI
    70: The element type "blank" must be terminated by the matching end-tag "</blank>
```

```
◇ equations-quadratic.oqmath (/Users/paul/activemath/content/mathebrHT...
   $a*x^2+b*x+c=0$ with $¬(a=0)$, compute its discriminant
     <p style="cm">$Δ = neg(b) + 4*a*c$</p><blank>
   If $Δ≥0$ then the quadratic equation has two solutions:
   <p style="cm">
     $x = ( neg(b) + √( Δ, 2 ) ) / ( 2*a )$ and
     $x = ( neg(b) - √( Δ, 2 ) ) / ( 2*a )$
   </p>
   If $Δ&lt;0$ then there is no solution.

   </CMP>   The content of element type "blank" must match "EMPTY".  Children of type "ENT
```

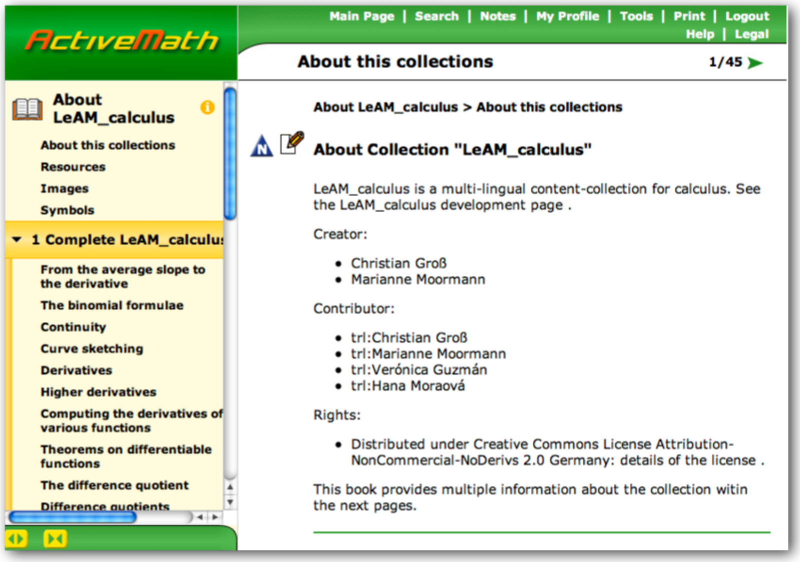
Figure 3.6: The error-list of jEditOQMath presenting both reference errors and validation errors.

- finally, references are validated: each reference in sources that are processed by the reload described below is *absolutized**, as explained in Chapter 4. This is done by verifying locations of reference targets. If this fails, an error is displayed to the user. This validation ensures that the relations will be functioning and will, thus, be used.

All these processes can generate errors which are presented in the error-list pane as shown in Figure 3.6. One of the fundamental aspects of the error-list is to bring the user close to where one expects he will repair the error. Sometimes this location is wrong, for example a dangling reference could be repaired by either the source containing the link or by inserting the content element target of this link. A more advanced management of these errors could be started as a new research direction. First hints can be seen in [Jed10].

The reference validation, as well as the transformation of the content along the QMath notations is done by the build process. This process is triggered from jEditOQMath's *Ant Farm* tab: the user selects the build-file of the collection and runs it pressing a button. This process is responsible for the **transformation** and **upload** of the content in the authoring ActiveMath server.

The transformation converts the OQMath documents to OMDoc documents, replacing all text-sequences between $-sign into an OpenMath* expression based on the notation-definitions which are provided in processing instructions –fragments between `<?QMath` and `?>`. Generally these processing instructions consist of references to other notation definition files. A second transformation is the automatic production of a collection-information-book, an ActiveMath book (see Chapter 2)



About LeAM_calculus > About this collections

About Collection "LeAM_calculus"

LeAM_calculus is a multi-lingual content-collection for calculus. See the LeAM_calculus development page .

Creator:

- Christian Groß
- Marianne Moormann

Contributor:

- trl:Christian Groß
- trl:Marianne Moormann
- trl:Verónica Guzmán
- trl:Hana Moraová

Rights:

- Distributed under Creative Commons License Attribution-NonCommercial-NoDerivs 2.0 Germany: details of the license .

This book provides multiple information about the collection witin the next pages.

Figure 3.7: The first page of the collection information book for the collection *LeAM_calculus*.

listing all the content items of the collections and all the resources; this book is the easiest way for an author to see newly authored content appear but is not expected to be of use to learners. Other authors may also use it to get an overview of the content of a collection. A sample of such a book is depicted in Figure 3.7.

The build process is concluded by a **content-storage upload**, a call to the authoring ActiveMath's content storage to reload all changed files: this process mostly makes them absolutized* and indexes the content. It is described in Chapter 4. It concludes with a reset of the caches for the indicated items (see [ULWM04]).

Having run this process, our author can now scrutinize his updated content within the Authoring ActiveMath Server – similar to how a sculptor steps back to observe what he is sculpting. This reload process is part of the authoring routine.

3.4.4 Interactions with the ActiveMath Platform

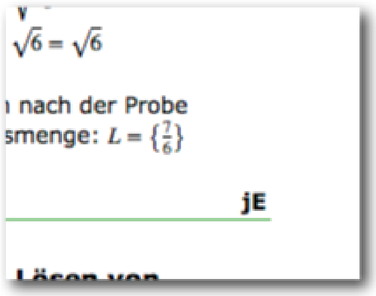
The previous section has described the editing and transformation processes operated on the content sources to yield a set of OMDoc files that ActiveMath can load.

Our author can now use his web browser to inspect what the resulting learning environment usage will be like. As often done on the web, most of the web pages delivered by ActiveMath pages can simply be reloaded to see the effect of the content changes. A few examples of reloadable pages are the following[5]:

- pages of a pre-recorded book are addressed by page number and book-identifier, a reload will display the updated content
- search queries are stored with the user-history, a reload will do the search again, on the updated set
- an interactive exercise window carries all the inputs in its URL; a reload will reproduce each interaction
- the course generation can be run again to obtain a book taking advantage of the changed items

One should note that these web-pages are quite diverse in nature but this is just a sample of the multiple perspectives, under which an author might want to proof-read the result of his authoring activity. This diversity has justified what is currently felt as a limitation of jEditOQMath: the lack of an immediate *preview of effect of the last authoring action* following a reload. I believe the wealth of accessible perspectives is exactly what an author is after when authoring and that the manipulation of ActiveMath in order to reach that perspective must be easy for him since it should correspond to the learning experience that he expects his students (or other learners) to have.

Actions back from ActiveMath into jEditOQMath also exist: I have described above how drag-and-drop references can create various inputs; another such action is the appearance of the jE button as pictured on the right: it allows the author seeing a piece of presented content to invoke jEditOQMath at the place in source where this content-item is.



Two more tools are available in ActiveMath for the authoring workflow, which I will describe elsewhere: the symbols-presentation-tool to explore the notations used by the server to render the semantic formulæ encoded by the author (see Chapter 7), and the remote reload facility which allows a web-browser to trigger an update and reload of a content collection (see Chapter 6).

[5]It appears that this requirement is not yet fully implemented by all components of ActiveMath for which compatibility to authoring tools has not been a requirement; as of this writing, the aim is to gather URLs that offer reloadable views imitating a state reached by students. For such components as the course generation this has been incomplete because the course generation creates a new book every time. Browsing the issue tracker of ActiveMath may give a grasp of this `http://jira.activemath.org/browse/AMATH/component/10025`.

3.5 jEditOQMath Technical Foundation

In this section, I review the most important technical choices that have made the implementation of jEditOQMath possible. jEditOQMath is a software written in Java* and built by the Maven 1.1 tool. Instructions to obtain it and build it are at `http://www.activemath.org/projects/jEditOQMath/`. jEditOQMath is delivered open-source mostly under the Mozilla Public License (MPL*) with some parts under the GNU General Public License (GPL*); see the web page to obtain the details of these licences and their coverage. It uses many libraries of the Apache Software Foundation and the jEdit project.

3.5.1 Ease of Installation

A first and foremost technical achievement of jEditOQMath is its ability for a fast and cross-platform installation based on the JNLP* deployment and update mechanism. This allows users of a *commonly installed* desktop computer to start the editor and the building process running in just a few clicks starting from the web-page. Implementations of this standard sometimes provide all the necessary bits so that the jEditOQMath application is called when an OQMath file is double clicked.

Many JNLP implementations, however, are not fully bullet proof and an alternate installation procedure is also available – a simple double-clickable application. jEditOQMath consists mostly of jEdit's installation, all the jar dependencies, and the installation of the QMath executable, being written in `C++`. The detailed components list is provided in the Appendix A.

The motivation for the Java*-web-start deployment is two-fold: Firstly, installation instructions, which are common in academic software, can be extremely fragile and wrongly executed and lead to long troubleshooting sessions; secondly, an ability to upgrade is very important in order to easily deliver new features and bug-fixes discovered along the research-oriented development process which characterize the tools around ActiveMath.

3.5.2 OQMath

The OQMath tool is a simple wrapper for the QMath processor [Pal06] that creates a QMath document from the notation processing instructions (between `<?QMath` and `?>`) and the mathematical formulæ islands (between `$` and `$`): it gives it to the QMath process, and collects the output formulæ, replacing the formulæ islands by their reformatted OpenMath* outputs. Having performed the replacement of the islands, one obtains an OMDoc document that is ready to enter into ActiveMath through its storage mechanism.

The reformatting identifies particular patterns to allow constructs of the QMath processor that I felt were missing at the time (such as the identifier of a term). Most importantly, the reformatting removes the indenting so that the formulæ in OpenMath* takes as many lines as the input formulæ does. This measure ensures that a line-number in the OMDoc document corresponds to the same line number in the OQMath source.

3.5.3 Location Tracking

Matching line-numbers allows feedback on content to be expressed by line-number. This expression practice appears less structured than XML*-based tree-coordinates but is considerably better supported: all contemporary parsers provide it. It can also care for more errors (in particular the well-formedness errors). The location information (by line-number) is carried until the indexing storage so that all reference errors can be issued by line-numbers. This allows error messages to be presented to authors who can move quickly to the faulty spot and act, when they see content displayed in any component of ActiveMath.

The line-numbers of individual content-items are also part of the storage described in Chapter 4 and this allows the jE button described in Section 3.4.4 to order jEditOQMath to open a given identifier, then jEditOQMath to consult its associated content-storage to obtain the file name and line number where this item is, and to open the source file to the user at the right line. It should be noted that the content storage consulted is not necessarily the one of the ActiveMath of the jE button: the ActiveMath could be a remote ActiveMath, e.g. a shared authoring commons, while the author ActiveMath's storage is consulted to find the file location.

3.5.4 DTD-aware-XML-outputter

A technical ingredient that is fundamental to all refactoring operations is a DTD*-*aware* XML*-outputter. So as to be able to support standards, the parsing result of ActiveMath is based on the widespread SAX* standard for parsing which provides the full XML information-set but does not offer a fully detailed model of the XML source (e.g. it omits to say that an attribute was present or is *implied*, that is, only tacitly set by the DTD). This has the consequence that a normal re-output of the parsing result is considerably larger than the corresponding source. Such a re-output, however, is necessary for refactoring: for instance, an operation that shows the result of the metadata* inheritance operation as described in Chapter 4 needs to modify the metadata element in several places and thus needs to re-output it. We have consulted the widespread expectations and it seems that it was acceptable to not have the complete freedom in XML-indenting in the case of OQMath documents. Instead, a DTD-aware outputter is used, which outputs an attribute if and only if this attribute differs from the default value in the DTD. This outputter is used in all refactorings and authors seem to accept its limitations.

3.6 Direct Manipulation for Authoring: WYCIWYG

In this section I propose to review how direct action, one of the principles of human computer interaction coined by B. Shneidermann in [Shn83], applies to authoring of e-learning content. It is based on a later analysis of direct action by D. Frohlich [Fro96].

Direct action refers to the immediacy of the *effects* of the actions of the user on the data. It has most often been used in a comparison between console interactions (such as a Unix shell) and visual paradigms (such as a desktop of files and folders). The analysis of Frohlich in [Fro96] is more nuanced and summarizes the results and analyses of experimental investigations following the direct manipulation proposal of [Shn83]. Among the lessons learned are two claims [Fro96, p. 475]:

- Visualisation of output data seems critical to the benefits of direct manipulation
- Visualisation of input data may be less important

The authoring approach of jEditOQMath follows these principles: the input interactions are done by the manipulation of file objects and textual sources with effects visible on the resulting learning platform as soon as possible. Similar practices can be found in user interfaces for the TeX layout processor, as I have described in Chapter 1: the user's objective is the realization of the resulting view but his actions are carried out on the source.

This approach is opposed to the WYSIWYG* approach, which requires the editing interface to be the same as the *target interface*. This is doomed to fail for authoring learning content on the web: in many senses, there isn't a single *What You Get*, but a plethora of perspectives, each of which some interest of its own when authoring. In ActiveMath, a simple example of this heterogeneity is the editing of the metadata* of the items: it can aim at the exploitation of the tutorial component, at particular search results one expects learners will exercise, or simply at the correct rendering of the content in a book view.

A more modern characterization of direct manipulation has been proposed – the *instrumental interaction* as described, e.g., in [BL00]: this approach describes the users as manipulating instruments, which are mediators (the word *transducer* is used there) to data objects operating commands on them and providing feedback to the user. The source editing paradigm can be analyzed by this method: the editing of the source is a manipulation of the *concepts representing them* which the ActiveMath learning environment interprets and displays. The source is a representation of the content that the jEditQOMath instrument offers for view and manipulation; another instrument of content manipulation is the authoring ActiveMath as described in Section 3.2. This instrumentalization, the process of acquiring the usage of the instruments to obtain intended effects, is particularly visible in the input of mathematical formulæ based on the QMath notation definitions. The authors' role is not to provide the QMath expressions but the OpenMath* expression and practice has shown that it is easy for authors to understand the mapping of QMath expressions to OpenMath objects, but that QMath expressions cannot be taught alone: they need to be explained with the OpenMath target in mind (e.g. with an understanding of the meaning of each symbol used).

An important nuance about the instrumental interaction approach is that the jEditOQMath instrument is relatively limited in its feedback (it manipulates the XML* source) but that another instrument allows a richer feedback. Moreover, this instrument is going to be the same tool that the learners will use. This view of the usage of instruments seems consistent with the instrumental genesis view of [RB03] and [GT08a] which considers the instruments as mediators, which also evolve along the usage which gives them their meaning. In particular, jEditOQMath seems to offer considerable freedom for the evolution in appropriation of the editing tools so as to best fits an author's usage.

I propose to name the paradigm implemented in jEditOQMath **WYCIWYG**: *What You Check Is What You Get*. The idea is that the editable authoring objects are manipulated in diverse fashions but the target experience that is being authored for is only obtainable by checking-routines within an environment that is close to the delivery environment. The WYCIWYG paradigm is classically found in online authoring where the view of an authoring tool always needs to be completed by a preview (e.g. using DreamWeaver*, Confluence*, or MediaWiki*). The WYCIWYG paradigm is rooted in the cycles between editing and previewing, which are similar to the cycles of writing and reflecting that M. Sharples described in *How we write: Writing as Creative Design* [Sha99].

3.7 On the Advantages of Plain Text Editing

Even though the plain-text nature of the source has often been criticized for its apparent technicality, it has important advantages which I summarise below.

Text fragments are **communicable** over emails and web-based forums. This is of a particular importance for purposes of documenting, learning, and discussing about ways to write content: a simple copy-and-paste allows a mail or tutorial to indicate the writing practice: to request help about it or to explain it. In comparison, visual editing tools require a much more elaborate language to describe the inputs, one that is often too imprecise so that videos of the screen are needed. Both of these methods have been used in documenting authoring with jEditOQMath as described in Chapter 9.

Another important facet of using text-sources is the **native support for merging** by contemporary versioning systems. To our surprise, this aspect has often been used in authoring even though a merge risks corrupting several aspects such as XML* structure, grammar, and reference. In practice, these corruptions have almost never occurred.

Finally plain-text **editors** are very **numerous** and the choice of keeping text has allowed several users to keep their favorite editors but still invoke many of the facets of the authoring system.[6]

3.8 Comparable Approaches

There is a wide range of tools being used for authoring mathematical content. On the low interactive learning side, one sees the usage of TEX, MS Word, MS Power-Point, or even Adobe Illustrator: these tools all focus on the delivery of static documents in an e-Paper form, hence their result can summarized in a single view, and they can apply the *What You See Is What You Get* (WYSIWYG*)paradigm. Active-Math's exploitation of the documents is considerably more interactive including the possibility for reorganization, adaptive course generation and the copy and paste of formulæ. Hence the authoring requires a different paradigm.

On the intelligent authoring tools side, several research-level authoring tools are presented in [MBA03]. Most of the descriptions there are descriptions of the knowledge models while, in this chapter and this thesis, we focus on the description of an authoring workflow.

The REDEEM learning environment, similarly to ActiveMath, allows authors to organize the content annotations so as to produce particular sequencing. REDEEM's user-model is very simple with authors quite in control of it while ActiveMath's is more extensible, as it can be enriched by combining multiple external content. One of the lessons learned from REDEEM's authoring tools article [AMG$^+$03, p. 229] is that *authoring tools will be more useful if they easily support progressive authoring*.

One of the model adaptive tools, the AHA! environment, supports an integrated authoring environment [BSS$^+$07] also based on files access with the knowledge layer almost separated from the content; the adaptive behaviours are, for most, authored within (XHTML*) markup enrichments. ActiveMath's content embeds both knowledge and content layers in a single layer which makes it easier to combine content collections and manage them. Instead of having a dedicated knowledge editor, jEditOQMath eases up the input and maintenance of references.

A tool sometimes considered close to ActiveMath, the Connexions authoring commons also works on content with semantically encoded mathematical formulæ. As far as we know, the strengths of Connexions lie in the community orientation of the sharing mechanism, while the authoring practice seems to be also based on source editing (see [Sch]). That source editing, though, seems to be only supported in the formulæ input and not, even, in syntax coloring. The fact that the

[6]The publish scripts are known to work with most editors, see `http://eds.activemath.org/en/jeditoqmath-offline` for a documentation.

Connexions' documents are aimed at an almost *static* delivery is considerably different than the expected usage of ActiveMath; hence the need for verification is considerably larger and thus supported.

The Assistments Builder project described at [RPA+09] is an authoring tool aimed at editing relatively simple interactive exercises by regular teachers. The usage features described in their paper seem to show a clear WYCIWYG* cycle, at least at the level of an exercise; beyond this cycle, the Assistments Builder's users are expected to be the same persons as the teachers and several tools are provided in this builder to track the students' progress and receive comments to enhance the content. The authoring experience of Assistments Builder claims to be easier to learn for teachers, and indeed it seems more visual, although the introduction of randomizers seem to present an editing experience at the level of the source authoring. Similarly to the Connexions' service, the Assistments Builder seems to be aimed at a single server: as a result, such tools as the global search tool or the tutorial component cannot exist and, indeed, the sequencing in Assistments is limited to hand authored links in predefined sets.

Very similar remarks seem to apply to the CTAT authoring tools [AMSK09]: the preview and check cycle is there as well, using the debug and run functions of the integrated development environments, and sequencing is manual only. CTAT authoring is considerably more visual than jEditOQMath but, at the same time, this constitutes its weakness: except for feedback detection it is only visual.

Compared to almost all of the softwares mentioned above, ActiveMath differentiates itself in the semantic nature of mathematical formulæ and the expected interoperability for (parts of) them. This feature forces the careful authors to check the input of the mathematical formulæ that learners might think of using.

3.9 Conclusions

In this chapter, I have introduced the central authoring place: the source editor and how it is integrated within an authoring workflow. I have named the workflow WYCIWYG to highlight the focus of authoring: the editing actions towards a target learning experience which the author checks within realistic settings via his Authoring ActiveMath Server. The jEditOQMath workflow typically follows cycles of editing within the editor and previews in the learning environment.

A fundamental requirement of this workflow to the **learning environment** is to allow **incremental changes to be previewed iteratively**. This requirement is important for the user interface of the learning environment, it is partially satisfied for several aspects of the learning platform such as book browsing but some other features could do it better. Addition of a collection in the authoring ActiveMath needs a lengthy re-indexing. It is very hard, thus far for an author to experiment with the user-model of another user. Finally, updated interactive exercises often need to be re-run to explore the result of a change. Although not fully feasible,

handles for authors to regain a perspective that learners may obtain in the learning environment should be strived towards: the web on which our learning environment exists is a natural support to this end since each web-browser can reload a web-page – provided the location is properly adjusted, the reload action should be just right to re-obtain a preview of the same commands' results.

The workflow is not novel. For example, it is common in HTML* authoring practices or in Wiki editing practices (as described in Chapter 1), but it has been rarely highlighted in literature about authoring tools for learning environments. This literature is, for most, concentrated on the editing interfaces or underlying knowledge representation as examplified in most papers of the survey book [MBA03].

A generalization of the WYCIWYG paradigm arises when different persons with different competencies are involved. For example, we can picture a team involving an encoding person, a didactics and domain expert, and a learning environment usage expert. Collaboration scenarios of this nature are most common in multimedia agencies but are not so commonly studied or applied to authoring tools literature. The book [LZC08] is a start of such description in the corporate learning world.

Planting roots of the authoring activity in the learning environment has the potential to transform consumers of the ActiveMath platform from intensive users, through active sequencers of content with the assembly tool (see Chapter 2), into authors. It has been shown in [Ben06] that participation to communities of open-knowledge developments can only happen if tiny contributions are possible: I contend that the integration in the learning environment is a critical enabler of tiny contributions, the other part being the sharing practice which I describe in Chapter 6.

3.9.1 Lessons Learned

The fact that jEditOQMath involves the edition of XML* s sources shocked newcomers and continues to do so except those with a source-authoring experience (such as an experience with TeX or WikiPedia). The various historical reasons that have lead to this have been presented in Section 3.3.1. However the editing tool is, by far, not a complete authoring tool. In this section I distill the functions that have turned the jEdit text editor into an authoring tool for ActiveMath. I expect these features to be close to requirements for the implementation of authoring tools which would allow freshmen to start authoring with almost no learning.

- **short reload cycles**: are certainly the core of the jEditOQMath workflow and are justified by the explorative nature of the authoring work I have steadily observed. More variations can be offered here, from one click builds to in-place-editing: most important seems the wish of attainment of the real learning experience the learners will encounter.

- **travel back**: conversely to the above, facilities to navigate from the content delivery interface to the source and being able to edit right-away, seems to help the authors observe the authoring process from the delivery side and act on it from the authoring side. I have experienced that several instances of the learning environments are commonly used when authoring, for example the one of neighbours or of demonstration servers. Being able to communicate fragments from an external environment to the currently authored content can also be useful.

- **error reporting**: freedom of authoring is useful for the authors' organization but feedback on possibly erroneous content is fundamental for a trustable authoring experience. Lack thereof can only be substituted by a long and delicate proofing at the destination (proofreading, *proofclicking*, ...).

- **content management with tangible objects**: authors know where is what in their authoring sources and they can easily open the right editing place to continue work or do a particular fix. The use of files in jEditOQMath naturally provides such facilities as grouping, backups, email transmission, time-tracking, exchange between collections, versioning systems... Such facilities may be otherwise lost and (probably partially) reimplemented.

- **communicability of the editing objects**: in order for authors to learn to author, or to discuss their authoring activity, it may be desirable to transmit part of the editing objects in electronic form. jEditOQMath, being based on plain-text, allows this easily. It seems that visual tools may require to resort other artifacts such as the description of the sequence of clicks which is error prone.

- **paste from the wild world**: maybe one of the simplest ways to start encoding content from outside is to *just let the computer do it*: a method to create a first approximation of the authoring input as a translation from content in other encodings can prove useful, letting the author enjoy the result in the learning environment and subsequently improve its source.

Chapter 4

Storage of Content in ActiveMath

4.1 Introduction

The content of ActiveMath is encoded semantically in XML* files. Components of this learning environment consume the XML in a basic manner, one item at a time, and need to be able to load these items efficiently.

This chapter describes the interface between the content files that the authoring tools edit and the learning environment. The interface is in the form of a storage component that is fed by authored files, and offers a uniform access to the learning environment's components.

The content storage described in this chapter makes it easy for an author to manage the content: to find, to re-use, to keep readable, and to input. This ease is enabled by four patterns which are likely to be generalized to other authoring practices: the organization in content projects, the aggregation by reference, the management of references in groups and imports, and the inheritance of metadata*.

4.1.1 Outline

In this chapter I describe briefly how the components of the ActiveMath learning environment expect to consume the OMDoc documents (Section 4.2). The approaches I could observe to solve the problem are then covered (Section 4.3). My contribution, the ActiveMath content storage, is then described in Section 4.4 by its content organization approach, its reference scheme, the set of methods it offers to the other components, and its central implementation. Section 4.5 follows with a description of the many content-storage-like implementations followed by a few facts about the performance reached. The implementation enables authoring practices described in Section 4.7 to be considered as authoring patterns. This chapter closes with a look at comparable authoring tools approaches and open research perspectives.

4.2 Mission of the Content Storage

The content storage is an interface between the files that are produced by the authors and the remaining components of the ActiveMath learning environment.

As explained in Chapter 2, the content that authors produce for the ActiveMath learning environment is encoded in the OMDoc language [Koh06] which is an XML-based semantic-oriented representation of mathematical documents. The OMDoc language provides a way to encode documents containing fragments with a mathematical role and with an identifier. A document may contain one or thousands of content items, similarly, a content item may be one line big or several pages big. Content items are stored in files suitable for XML processing.

4.2.1 Identifiers and References

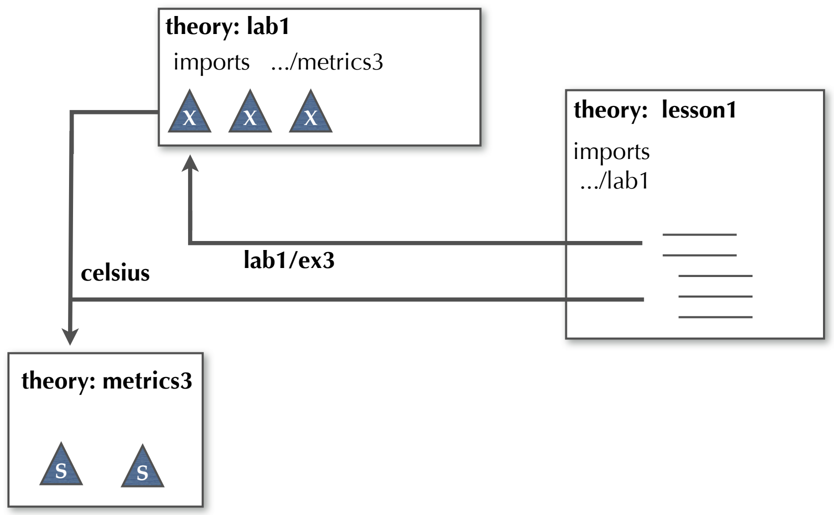
Identifiers in OMDoc are expected to follow the development graph paradigm introduced in [MAH06b]:

- content items can be grouped within *theories* (represented by a `theory` element): identifiers are only expected to be unique within the theory.
- within a theory, references can be expressed using `imports` which point to an external theory. Doing so allows to reference the items of this other theory as shortly as the items of this theory.

The references described above, especially the use of the imports elements, allow great freedom in reference management. However, in order to identify an item pointed to by a reference attribute – that exists within a content item, which itself is nested inside a theory – all imports need to be followed and a verification of the content items existence needs to be performed. I call this operation the *resolution* of a reference.

An example of resolution is depicted in Figure 4.1: in this example, the reference attributes' values are short: they are depicted in the arrows' labels. This graphic shows well the work of the resolution process: one needs to crawl through the imported theories to actually find the item pointed to by such a reference value as `celsius`.

ActiveMath's functions described in Chapter 2 (such as the display of content, the play of exercises, the planning of courses, or the updating of the user-model)



1

Figure 4.1: An example of a simple imports situation: the references indicated in the arrows are the content of the references attribute.

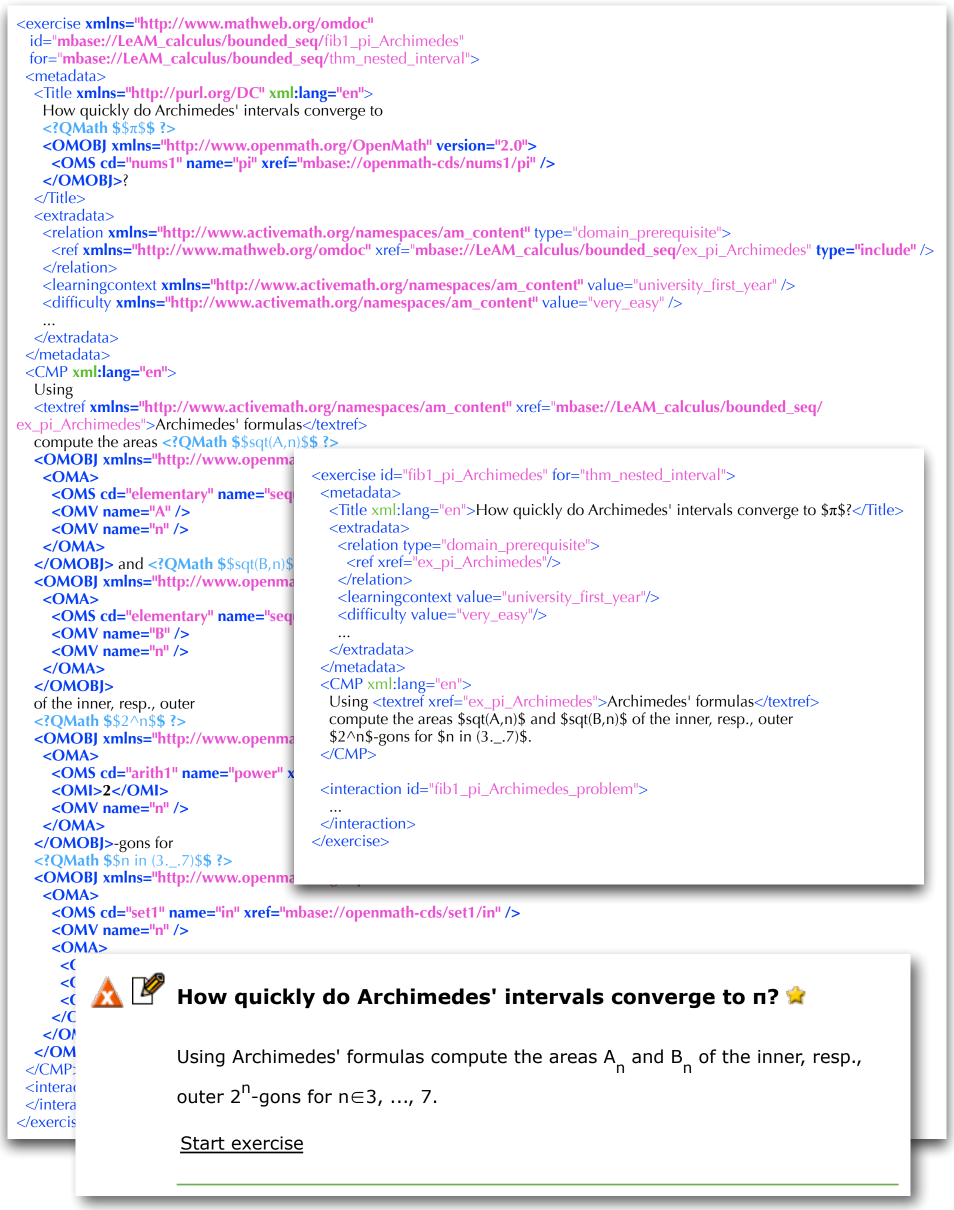
should not be concerned with resolution as they would, otherwise, need to load all files following imports so as to check for existence. They need to be given homogenous references and identifiers which, ideally, are compared using simple string comparison.

4.2.2 Fragment Extraction

The ActiveMath components only see the OMDoc content as elementary content items: a fragment of an OMDoc file addressed by an identifier. The storage should support this fully by only delivering the content items in isolation with all references resolved. I call this process the *decontextualization* of the content items: the items can then be used outside of their context, be presented alone in the search tool, or be combined with others within a book as chosen by the tutorial component, by a teacher, or by a student, ...

Here are a few fragment extractions by components of ActiveMath:

- the presentation system reads the `metadata` and the textual fragments of each content item of a book page, as well as the title of each symbol occurring in there
- the exercise system reads the entirety of the content items
- the search results' reads the `Title` of each search result and the `metadata` of the item being presented



```
<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 $$π$$ ?>
    <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
  <textref xmlns="http://www.activemath.org/namespaces/am_content" xref="mbase://LeAM_calculus/bounded_seq/
ex_pi_Archimedes">Archimedes' formulas</textref>
  compute the areas <?QMath $$sqt(A,n)$$ ?>
  <OMOBJ xmlns="http://www.openma
   <OMA>
    <OMS cd="elementary" name="seq
    <OMV name="A" />
    <OMV name="n" />
   </OMA>
  </OMOBJ> and <?QMath $$sqt(B,n)$
  <OMOBJ xmlns="http://www.openma
   <OMA>
    <OMS cd="elementary" name="seq
    <OMV name="B" />
    <OMV name="n" />
   </OMA>
  </OMOBJ>
  of the inner, resp., outer
  <?QMath $$2^n$$ ?>
  <OMOBJ xmlns="http://www.openma
   <OMA>
    <OMS cd="arith1" name="power" x
    <OMI>2</OMI>
    <OMV name="n" />
   </OMA>
  </OMOBJ>-gons for
  <?QMath $$n in (3._.7)$$ ?>
  <OMOBJ xmlns="http://www.openma
   <OMA>
    <OMS cd="set1" name="in" xref="mbase://openmath-cds/set1/in" />
    <OMV name="n" />
    <OMA>
     <(
     <(
     <(
    </C
   </OI
  </OM
 </CMP:
 <intera
 </intera
</exercis
```

```
<exercise id="fib1_pi_Archimedes" for="thm_nested_interval">
 <metadata>
  <Title xml:lang="en">How quickly do Archimedes' intervals converge to $π$?</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 $sqt(A,n)$ and $sqt(B,n)$ of the inner, resp., outer
  $2^n$-gons for $n in (3._.7)$.
 </CMP>

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

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 $n \in 3, \ldots, 7$.

Start exercise

Figure 4.2: An extract of an OQMath source of an exercise (prior to applying the OQMath transformation, its compiled and decontextualized form on the back, and its rendering.)

Figure 4.2 presents an exercise in 3 views: its view, its source authoring form and in how it appears in its decontextualized form. The differences are in bold: they are the formulæ which are converted to OpenMath*. The default values of the DTD* (referred to in the header of the document) are applied (e.g. the `type` of `ref` elements or the `xml:lang` attributes) and the references are resolved (e.g. the `for` attribute of the `exercise` element).

Finally, some ActiveMath components need to query the relations from and to the content items:

- The tutorial component needs to query all the exercises that are `for` a given definition.
- The search tool displays examples `for` any given concept and displays the examples of a concept.
- The user-model queries the concepts that the exercise is for when receiving report of an exercise step as well as the prerequisite relations.

4.2.3 Performance

It is expected that the components of ActiveMath make many queries to the content storage: the tutorial component needs to rapidly scan all available content around each concept and each dependency of that concept. The book-page presentation system needs to assemble between 10 and 20 content item and its metadata* for a single page delivery as well as all the symbol names. The user-model needs to query the metadata of many content items as well as the relations (in and out).

An estimate in the description of the tutorial component's mediator [KUM06] yields around 10'000 queries for a typical book-planning which justifies several layers of caching. Moreover, a set of content of the size of about 500 printed pages should be easily handled. Thus the performance of the content storage when delivering the fragment and relationships is an important issue.

4.2.4 Authoring Services

The content storage functions should offer services to the authors so that they can let the content evolve and see the evolution in the learning environment. I propose that the content storage can *reload* from OMDoc files that were just modified or generated. The reload process should, as long as errors are still possible, report errors that help the author understand the necessary tasks in order for their input to be completely understood.

4.3 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: based on a set of source documents, they organize a storage which is only understandable by themselves. Such storage includes, for example, an extraction of all the links so that traditional wikis are able to indicate *Wcalledhat links here*. The internal representation, often in SQL databases, is rarely discussed. It is part of the internals of the systems, but it is crucial to the functions of the system.

The decontextualization requirement is not to be understood as a special requirement of the approach of manually editing XML* files or even simple source formats: 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 to resolve URLs from relative to absolute or to style the rendering based on user's preferences. This decontextualization is the natural transition from a context of authoring to a delivery environment where such readability advantages as relative links become unnecessary and impossible.

A simple organization of OMDoc files is in the default OMDoc software distribution from `http://www.omdoc.org/`. It showcases nicely the multiple exploitations of OMDoc contents: the 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 to serve individual learners: in particular the assembly of pages made from content items of different sources and the query of incoming and outgoing relations. Such functions can be provided with this simple processing model, but their operation does not scale to sizes expected for a classroom delivery. As a result, most of the references found in this repository do not use the facility of imports but the simple facility of an identifier within a precise file. To our knowledge, attempts are underway in the OMDoc project to enhance the reference schemas within the MMT effort [Rab07], and it seems unavoidable to maintain an index of all names and imports for their resolution to be performed.

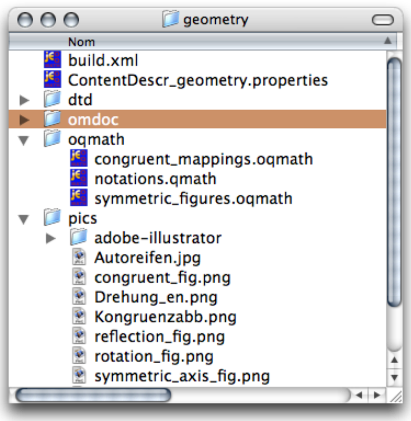
The idea of an intermediate layer that resolves the URIs has also been applied in HELM's getter, which is slightly documented at `http://helm.cs.unibo.it/software/getter/`. The Getter's aim is to serve as gateway to access multiple sources of contents; it delivers XML* documents whose internal references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

A natural family of products that should have played a role in our situation is that of XML databases. Unfortunately, the heavily hand-authored nature or the relatively complex files' structure have always prevented such a tool to be a viable solution. Attempts were made with dbXML, Berkeley DB's DBxml, and eXist, all supposed to be best of breed of their generation, but all failed at some scalability issue: document size too big for a string parameter, document depth too big for the normal optimization, too slow requests...[1]. Candidates which are intended work on a large scale such as TopX [TSW05] have been considered and somewhat experimented with, but they have been discarded because their *easy and free deployment* for a school or an author has not been sufficient.

4.4 The ActiveMath Content Storage

Above, I have described the mission of the storage and the existing libraries and approaches to help in the implementation of the storage. In this section, I describe the organization of the storage in detail: how content files are organized by authors, how the components of ActiveMath see the content, and how the current storage is implemented in order to fulfill this mission. In a later section, I shall describe the alternative implementations.

4.4.1 Content Collections as Units of Authoring Management



```
build.xml
ContentDescr_geometry.properties
▶ dtd
▶ omdoc
▼ oqmath
    congruent_mappings.oqmath
    notations.qmath
    symmetric_figures.oqmath
▼ pics
  ▶ adobe-illustrator
    Autoreifen.jpg
    congruent_fig.png
    Drehung_en.png
    Kongruenzabb.png
    reflection_fig.png
    rotation_fig.png
    symmetric_axis_fig.png
```

The mission above does not state how OMDoc files are to be laid within a directory structure, or within a web-managed space. And indeed the repository at `http://omdoc.org/` has taken quite various strategies to layout the files. One of the important missions, however, is that of enabling the re-use of content and thus propose best-practice in the organization of files.

I propose to assemble contents by *content collections*. Content collections define projects of authoring and collect in a filesystem's directory all files relevant to shared authoring. A file-manager view of a simple content collection is on the left. Content collections have a public-identifier (`geometry` in this case) and are introduced by a content-descriptor, a file in the base directory whose name starts with `ContentDescr_` and ends with `.properties`. The descriptor provides the references to the directory of OMDoc files to be loaded (the `omdoc` directory) and the directory of web-accessible multimedia resources (the `pics` directory). Content collections generally are self-contained with all satellite files included; among them the directory of DTD* files

[1]The original intent was to leverage an OMDoc storage called MBase, described in [FK00], whose symbolic query capabilities were rich. Its development, however, has been slow with late releases never offering sufficient performances.

is included. It specifies the grammar (and default values) for the XML* files that refer to it, hence it defines the model the content collection is following. Carrying the model along with the collection allows the authors of the collection to decide when to upgrade to a newer model, being careful of doing the necessary upgrades in a compatible and pedagogically sensible way instead of just depending on the server version's underlying model.

The interest of content collections lies in the proper definition of the re-usable building blocks: a content collection is a well defined set of files that can be loaded in an ActiveMath server and, maybe even more importantly, can be further authored. The content storage sees collections as a set of files to be (re)loaded. Generally, this reload is triggered by a build-file that is also part of the collection. Working on a collection generally involves using the same notations, the same DTD* and the same (global) metadata* as everyone: this enables authoring to be shared. I refer to Section 6.4 for a more thorough description of the advantages of content collections for the purposes of re-use and appropriation.

4.4.2 Collection-based Identifiers: the mbase:// URI-scheme

So as to enable the notion of collection and references between collections, independent of the files layout, the proposal was made to have references be made of collections, theory-names, and identifiers. I made this proposal on the OMDoc mailing-list in 2003; it never became the first choice of the OMDoc's main author but complete proposals that incorporated the notion of imports only came recently (the first emerging in [Rab07]).

In ActiveMath, thus, absolute identifers are of the form:

```
mbase://collection/theory/name
```

or

```
mbase://collection/name
```

where `collection`, `theory`, and `name` are arbitrary URI-fragments without slashes. The reference `mbase://collection/theory/name` denotes the element whose `id`-attribute is `name`, within a theory element whose `id` attribute is `theory`. This sits within any OMDoc file of the content collection whose public-identifier is `collection`.

Relative references work with this scheme and can be influenced by the usage of the `imports` children of the `theory` element:

- Any `theory` element can have `imports` children. They point to theories in the same collection or in other collections; these theories are called *imported theories*. Imported theories may grow transitively: importing a theory `A` that imports (globally) a theory `X` will import the items of theories `A` and `X`.
- From the item whose absolute identifier is `mbase://collection/theory/name`, a reference made of a single name `singlename` refers to an element whose identifier is `singlename` within the same theory or, if not existing, within any imported `theory` (in order of preference from top to bottom).
- From the same item, the reference `sometheory/somename` indicates the element of `id` attribute `somename` within the theory of `id` attribute `sometheory` in this collection, or an imported theory of `id` attribute `sometheory`.
- From the item whose absolute identifier is `mbase://collection/name` (i.e. not included in any `theory` element), a reference made of a single name `singlename` refers to an element whose id is `singlename` within the same collection and outside any theory.
- From the same item, the reference `sometheory/somename` indicates the element of `id` attribute `somename` within the theory of `id` attribute `sometheory` in this collection.

Although I acknowledge that a private URI-scheme is not an ideal choice, and indeed it is discouraged in [JW04, sec 2.4], there has been no complete alternative (until recently) that would also include the reference management flexibility of `imports` elements.

The mbase:// URI-scheme offers powerful management of references so that short references are written within the text and are fully resolved by the usage of the `imports` elements which provide the *répertoire* of reachable references.

4.4.3 The MBaseRef Contract

Which information is extracted from the content and how is it extracted? I describe this in a set of functions – a Java* interface called MBaseRef; this interface has been quite stable over the years. Its current version is documented in [LF03]. It contains the following functions which almost all take as parameter the absolute identifier of a content item and return, for most, a parsed JDOM* element:

- `getTextualContent` and `getFormalContent` extract the `CMP` and `FMP` children
- `getCommonName` to return the `commonName` children as well as the `Title` children of the `metadata` element
- `getMetadata` to return the metadata element

- `getChildren` to return the complete element
- `getForWhat` to return the resolved identifier referred to in the `for` attribute of the item
- `getType` to return the element name possibly complemented by the value of the `type` attribute (e.g. `omtext/elaboration`)
- `getIncomingRelations` and `getOutgoingRelations` to return the linked content-identifiers as well as the relation type for each relation
- `getCollectionsProvided`, `listTheories`, `listItems` which list the available collections, the available theories in that collection, or the available items in that theory
- `getLastModified` gives the modification date of the file containing the item
- `getOMDocPath` gives the file-name, path, and collection of the content item, as well as the line-number and column-number where it is located
- `reload` and `reloadWithErrors` requests the reload of all items within changed files in the indicate collections. The latter allows the remote calls to progressively receive errors reports as the reload happens; this is used in the build system triggered by the jEditOQMath (see Chapter 3).
- `previewMetadataInheritance` and `generateImports` are two author-oriented functions to help in populating and proofing the contents being input (see below).

Encompassing this set of methods in a Java* interface has allowed several implementations which I shall describe below. This set of functions is used by (delegates of) the many components of ActiveMath providing a uniform access to the OMDoc content, void of all the context-dependencies and independent of the underlying implementation (hence working both locally and remotely).

4.4.4 SLuMB, a Storage for Large Content Collections

The storage implementation is a component of ActiveMath. Similarly to all other components, it is configured using the properties* files described in Chapter 2 which includes the properties defined in any content-descriptor: a content-collection is loaded by the fact that its content-descriptor is copied (or symbolically-linked) into the `conf` directory.

The current implementation of the storage is called SLuMB (serial Lucene MBase). It is designed to handle multiple large content collections by indexing their content within two Lucene*-based indexes and a mass storage, and offer the MBaseRef functions by reading this index.

SLuMB uses two indexes: one for relations between the content items and one for the remaining data (the *small index*); the remaining data is made essentially of serialized representations of the answers to each of the possible queries following the simple *document* model of Lucene*: a set of key value-pairs associating names to values.

An exception is the so-called *mass-storage* which contains the responses to the queries for `getChildren`, `getTextualContent` and `getFormalContent`: the small index only stores the number in the mass-storage while the mass-storage keeps a file with that representation. The reason of this separation lies in the frequent mishandling of field-values in some versions of Lucene* that load them as strings entirely loaded in memory, while a handle to an input-stream would be sufficient. Loading in memory is fully inappropriate for several content types (for example the response to `getChildren` for multi-step interactive exercises is commonly more than 600'000 characters long, the response to `getChildren` for common theories is 2'000'000 characters); such a miss-handling is not restricted to Lucene and is commonly encountered in interfaces to SQL databases including the widely used Hibernate library for object-to-SQL persistence.

The SLuMB load process is a multiple phase process which operates the decontextualization so as to store in the indexes and mass-storage all the necessary fragments void of their context dependencies. The reload process only differs from the load process in that it only loads the changed files. It happens as follows:

- the list of loaded collections is loaded and evaluated against the already loaded ones
- for each collection being reloaded, the directories of OMDoc files are scanned so as to find the newer files that need to be reloaded
- for each file to be reloaded, the information recorded about it is erased
- a first full pass through the files to be reloaded is done by scanning the identifiers and imports which allows to record a first skeleton
- each file is then loaded as JDOM* document, and each item that has an identifier is processed:
  - XML*-parsing injects all the attribute default values (including namespaces)
  - all identifiers are made absolute according to the enclosing collection and theory
  - all references are resolved according to the possibly enclosing theory and its enclosed imports: this process requires that the skeleton of existing theories, identifiers, and imports be crawled through.
  - metadata inheritance is operated (see Chapter 8)

- After this process, the mass-fragments are stored in the mass-storage, the relations are stored in the relations index and the other fields are populated in the small-index.
- Finally, an `MBaseCollectionsChangedEvent` is fired to all listeners which triggers, for example, cache invalidation and the rebuild of the notations'-produced stylesheets if a notation was changed.

This reload process is the central operation of the storage after which only read operations are performed. Because it only operates on changed files, it can be used by authors to preview the results of their changes within their authoring-ActiveMath in less than a minute. The full load process can be fairly lengthy – hence should only be used to build fresh new installations. The lengthiness of this process is to be counterweighted by the excellent performance that one can obtain at reading time as documented in Section 4.6.

4.5 Multiple Content Storage Implementations

The MBaseRef set of functions described in 4.4.3 is relatively small so that different implementations have been possible. I describe them, first in a historical perspective, then along the different functions:

The content-storage component has a long history in the 10 years of ActiveMath development:

- The first implementation, in 2000, was based on a DOM-interfaced representation of the XML* documents created by Apache Xerces: the complete load in memory allowed easy access by identifiers of all the functionalities except for the backwards relations which needed cross-seeding. The loading time was long for relatively small content (20 minutes for about 30 pages of content); as a result the content storage was a server accessed by other ActiveMath instances over the XML-RPC* protocol.
- Moving to a representation based on JDOM* made it possible to divide the loading time by two; changing from one parser to another gave another division by two. The server approach remained.
- A rewrite followed around 2001 with the important mission of reporting reference errors in an author friendly way. The OMDocJDOM-MBase was born. It remained an in-memory database which could be accessed significantly faster, or used as a server. Several failed attempts followed.
- The first failed attempt was the usage of the dbXML then eXist databases. The code for eXist is still there but it fails with a slightly complex OpenMath* formula.

- The second failed attempt was that of taking advantage of A. Franke's and M. Kohlhase's MBase [FK00]: using it was the original plan but several technical impediments due to the platform difference delayed it. The released MBase code yielded too slow response times where the back-end MySQL saturated.
- In 2003, Shahid Manzoor started an implementation that imitated OMDocJ-DOMMBase but stored the fragments in a Lucene* index. Although stability took long to be acquired, that implementation became the recommended approach after one or two years with noticeable performance gains.
- In 2006, I started SLuMB, a complete rewrite of LuceneMBase, so as to avoid the memory load due to LuceneMBase still resolving the links on the complete in-memory-load of the OMDocs as JDOM* objects. SLuMB's multiple phases avoided this. This is still the officially recommended content-storage living inside each ActiveMath.

The Java* Interface nature of the MBaseRef set of methods allows multiple implementations and this has been extensively used:

- The first and most classical implementation is that of a **remote connection**, using the XML-RPC* method: all methods trigger an HTTP* request, with XML* fragments being passed as strings. This method is widely used by client tools such as the assembly tool or the concept-mapping tool (see Chapter 2). It is also used to invoke the reload mechanism within jEdit-OQMath. As explained above, this method is less appropriate to perform for such queries as `getChildren`. The remote connection client started with the very first implementation and is still being used. Current ActiveMath distributions make such an interface available at the path `/ActiveMath2/xmlrpc/mbase`
- Another wrapping implementation is that of a caching content storage that stores the last 1'000 requests's results returned by the content storage it wraps: this allows a subsequent equivalent request to fetch the result from the memory; in particular, this avoids parsing to create the JDOM* fragments. This content storage helps, for example, the query mediator [KUM06].

The reader is referred to the Appendix A for details on the availability of each of these implementations.

4.6 Performance

As explained above, the content storage is an important piece in the speed of the ActiveMath experience. Several memory bottlenecks that have represented issues in earlier implementations have been addressed as I described above. The resulting implementation has an enjoyable performance at reading time:

- As reported by [KUM06], in 2006, a typical course generation sends, without cache, 10'000 queries to MBase (and plans in 10s), and, with a cache of 2'000 results, 1'000 on the first run (plans in 5s) and none on the second run (plans in 0.8s).
- A normal reading activity of an ActiveMath server performs well: downloading the 500 pages book of LeAM_calculus* is delivered in HTML* in 2'30 that are done using 27s of pure content-storage time and 160'000 queries. This yields a mean query speed of about 5'800 queries per second.

A bottleneck remains in the indexing time: on some computers indexing of such a content-collection as LeAM_calculus* (now containing about 6 times 500 pages) can take as long as 30 minutes, but takes 2'30s on contemporary servers (Xeon 8-core 2.26GHz). The critical ingredients to avoid such a performance bottleneck is sufficient RAM (more than 1Gb) and a sufficiently fast hard disk (e.g. no network shares) to store the index.

Such measures may be related to the large number of files of the mass-storage described in Section 4.4.4. But this has never been a priority since the authors' normal workflow uses a simple reload which should take (depending on the amount of change) less than a minute before the content can be seen in the authoring-ActiveMath, including an error report.

4.7 Patterns of Management of Content

Above, I have described the technical infrastructure of the content storage based on its original mission and its evolution. In this section, I present the vision of content management that is enabled by the implementations described above. The management is described by patterns that are likely to be generalizable to other systems.

4.7.1 Re-usable Content Collections as Directories

The first pattern stipulates that content-authors group the contents in 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 implements this pattern by loading a set of collections and offering them to the consuming components. Each collection is a directory with a descriptor and an amount of content files.

The objective of the separation is not only to group common topics together but rather to group together a common set of practices and goals. Therefore a collection 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. Sharing a directory in a subversion server is a typical method which most 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 to maximize the common authoring 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. I refer to Chapter 6 for a broader discussion of the sharing practices.

4.7.2 Aggregation

Aggregation of content from several other pieces to create a larger work is the second pattern that is enabled by the content storage of ActiveMath and the presentation system's *book* paradigm, as described in Chapter 2. The latter is implemented using table-of-contents which are special content item that refer to the content items of their constituents to denote inclusion; the constituents can be of the same collection or another.

Aggregation is the natural complement to separation in projects, and together they allow rich re-use scenarios which preserve the long term development of content projects. They are discussed in Chapter 6.

Aggregation is authored by loading into one's authoring ActiveMath all content collections to be assembled, assembling them in a table-of-contents, and finally by checking the result so that it looks like a homogenous sequence. Such checks often leads to modification wishes which are enabled by modifying the local copy of the content collection. How often is the modification wished for? This question has no definitive answer; a study addressing this has been realized by M Lokar in [Lok98] and concludes that 80% of the math-teachers in the poll want adaptability of the learning resources but 10% actually will do the adaptations; it is probable that such amounts apply here.

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.

4.7.3 Management of References with Imports

The third pattern proposes that references between content items are kept as short possible in the middle of the content, relying as much as possible on import statements that expand short references into larger ones. This pattern aims at readability of the references, be they input in source text or in input-fields.

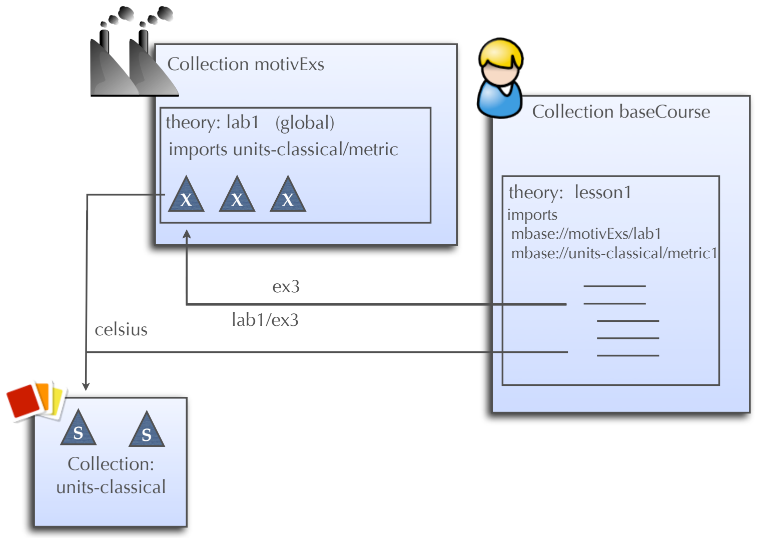


Figure 4.3: Short references with imports between items of various collections.

The objective of this pattern is to help with one of the particularly error prone parts of authored content which is the insertion and care of references. In OMDoc, references are numerous:

- the metadata* often contains references indicating such relationships as *is-an-exercise-for* or *pedagogically-depends-on*
- hyperlinks 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 (see Section 2.1).

Because of this large amount of references, it is necessary to keep references readable and verified at all times. 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 [MAH06b]: they propose to group (formal) mathematical objects within mathematical theories and to allow theories to be imported into other theories to extend the second; they 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 [Koh06]. The mbase:// URI-scheme described in Section 4.4.2 implements this fully for the purpose of managing references. Both the OMDoc language and Hutter and Autexier confer to this approach the semantics of categorical functors (such as [ML71]) which seemed to be too strong a requirement for authors. Therefore all documentations describe `imports` and `theory` elements as aspects of reference management.

The authoring best-practice 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 value as short as that of reference to a local item. Following example references described in Section 4.2.1, a typical reference situation would be that of Figure 4.3. Such short references as `celsius` or `ex3` become fully resolved thanks to the imports. This is to be compared to the references depicted in Figure 4.4.

Grouping the *binding statements* such as the imports in one place not only helps to keep the references short but also helps to follow the evolution: as a natural case of evolution, a content collection may have become revised by an author team different than the authors; in this case the collection name should be ideally changed. For example the collection containing the theory `lab1` may decide that a more modern collection for the units is required; their authors only need to change their imports elements from `units-classical` to `units-modern`. The change will be sufficient to be propagated to any importing theory.

References' correctness can be ensured by permanently validating them, as is done in the reload mechanism described above. This validation reports the line-number where unresolved references appear. This allows an easy "click and fix" approach.

However, a very common situation makes the input of `imports` elements rather a tedious task: when the theory and name pair suffices to identify fully the target elements. The usage of QMath notations as described in Chapter 3 typically generates `OMS` elements which, in the majority of cases, point to the normal OpenMath*-CDs content-collections which represent the widespread content-dictionaries of the OpenMath website. To avoid the burden of inputting such `imports` elements,

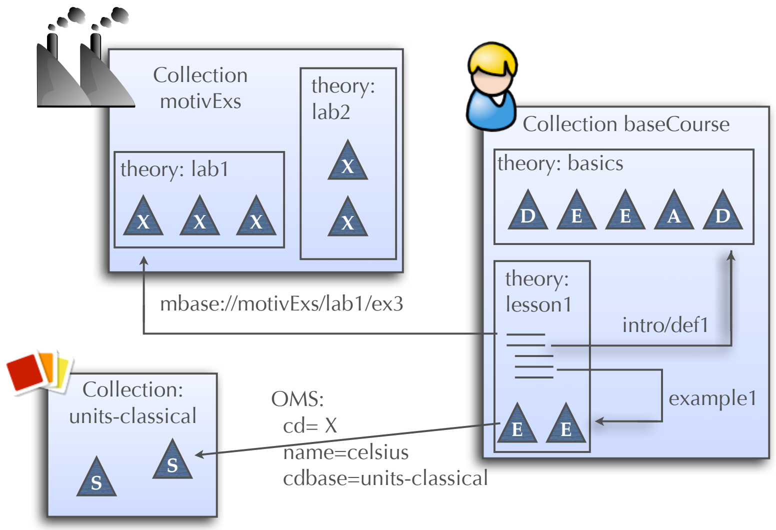


Figure 4.4: References between items of various collections tend to be long.

the content storage offers a guessing mechanism which is a function of jEdit-OQMath: *generate imports*. This function takes an OQMath file with its OQMath compilation and attempts a reload-with-guessing: any reference that is not fully resolved (for example an element `<OMS cd="arith1" name="divide"/>` without an `<imports xref="mbase://openmath-cds/arith1/"/>`) is guessed by the `generateImports` content storage function: all content items with same name within theories of the same names are searched and the corresponding `imports` element is suggested. In this example it could add

```
        <imports xref="mbase://openmath-cds/arith1/"/>.
```

The result is inserted aside of other `imports` elements as first children of the `theory` element thus becoming an almost transparent process for authors that wish to ignore the `imports`.

4.7.4 Manageable Metadata Through Inheritance

The approach above proposes to rely on the content of enclosing `theory` elements to provide missing information in reference elements. It uses the context of a content item to enrich it as is done in most language-based communication (e.g. when one refers to *the living-room* indicating the area of the apartment we're living in). The same context inheritance is proposed as the last pattern: most of the metadata* should be inherited from their enclosing elements. This is described in Chapter 8.

The content storage is responsible for operating the inheritance which it does at reload time. Supplementarily, the metadata inheritance can be *previewed* by requesting the inheritance-enabled metadata element of the item under the cursor – it invokes the `previewMetadata` function of the content storage.

Both of these patterns, that of imports-based references and of metadata inheritance, carry the *issue* that copy and paste of a source fragment may have unexpected effects such as resolving notations differently or bringing other target educational levels. I claim that such surprises are desired and normal.

4.8 Conclusions

In this chapter, the storage has been described as the main interface between the XML* content files that authors manipulate and the components of the ActiveMath learning environment. That interface is summarized in a short set of methods which allow the components of the learning environment to obtain a uniform access to the content items ignoring their context. This interface allows several authoring practices which strongly facilitate the management of content: the organization of content in directories of projects and the aggregation within books, the management of references by imports and the inheritance of metadata.

In this conclusion, I describe approaches that are similar, overall, to our approach and hint at further research investigations.

4.8.1 Comparable approaches

The practice of using import statements to bring a namespace within the current namespace is similar to the usage of the `import` statement in the Java* language. The contemporary development environments show several interesting approaches to interactively populate the sequence of imports while the code is being built by the exploitation of an index of classes. Three major differences exist however.Firstly, the import statements of Java are exploited at compilation time, whilst secondly the important elements of OMDoc are exploited by the ActiveMath content storage at loading time. It is not clear which practice is best but that of ActiveMath is clearly less predictable. Thirdly, the imports elements of OMDoc (and the development graph) are recursive while those of Java are not – this, again, contributes to a higher unpredictability but does contribute to a smaller set of imports and easier dependency management.

The recommendation to exploit directories of files as units of authoring work seems very common: all integrated development environments, including those used by the CTAT tools [AMSK06], apply the the concept of project as well as the concept of imported project. Sites in DreamWeaver* do the same. A few authoring tools such as the AHA! authoring tool [BSS$^+$07] are based on multiple files which, together create a learning experience but do not seem to speak of project directories. More generally, because a learning activity under the direction of an educator is almost always made of multiple materials, several models of learning resources sharing are composites: this is the case of the templates of Curriki [Kur08] and has been long advocated in the rigid model of SFoDEM [GT08b]. Multiple aggregations, as done by ActiveMath's content collections are commonly suggested but often lack being described as best practice.

4.8.2 Open Questions

Combining Remote and Local Storage In principle, the XML-RPC* remote connection method of content-storage access could also be used to offer the access to a content-collection from within another ActiveMath. This would provide an important way to *just try* a content collection before deciding to re-use it (e.g. include some of its items and see how they fit). Unfortunately, the current implementations do not support multiple content storages to be combined. A combining storage would need, at time of resolution, to access all other content storages, so as to check the existence of content items and, more importantly, to partially redo so when the new content collection is configured. This has not been explored yet.

Pre-Indexed Storage As we have seen above, the time taken to index is a bottleneck. Distributing indexing result may solve this issue. This would involve adjusting the SLuMB implementation to store one index per content collection, and to share this with versioning systems. It is not clear whether this would not cause more conflicts than improving and speeding up the process, and maybe a compilation approach which stores OMDocs in a complete decontextualized form may be easier.

Change Awareness 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 introduction of new notations may change the way things are rendered, changes in imported theories may trigger an error state in references that were correct before, new content items may change the behaviour of the course-generation or search engine and more changes may happen. 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 reload the necessary bit; this makes the changed notations to become visible almost right-away, for example. Similarly to [AFNW07] but impact can be considerably deeper than just notations with the need to consider relationships in both ways. Guidance to *see the changes* may be wished (for example *which page of content in my book has been affected by the reload? how different is my typical course generation now that I have this supplementary material*?).

Metadata Reasoning Finally, the reload process described here offers much space for the application of reasoning techniques so as to update diagnoses about the content. The current validation is relatively mild (DTD* and references only); the usage of ontological techniques, converting the metadata* into ontology fragments against which axioms can be verified would yield rich diagnosis possibilities which might empower the author feeling safe after he has addressed all issues reported to him. I view this as a larger knowledge representation problem within the future works of this thesis, Chapter 10.

Chapter 5

Getting Access to Mathematical Content

Introduction

Access to mathematical content items on the web can be achieved by several methods. The most prominent ones are *search* and *links*. Both methods are currently used for accessing parts of the content of ActiveMath. We describe both of them.

This chapter first introduces the learning and authoring situations in which content items are presented and searched for. This is followed by a description of the search tool's components and evaluation results. Finally, related research is presented. This chapter is an update of [LM06].

5.1 Usage Situations of a Search Tool for Learning

This section reviews common practices for mathematical knowledge management and indicates what ActiveMath is doing in this direction. I use the term *access* very broadly for the ability of a user to reach a given item, symbol, or formula. Accessing an item means to have it presented in a browser, to be able to reference it or to let other programmes download it.

The activity of searching mathematical texts or formulæ has the same purpose: obtaining access to a content item of interest by the formulation of an easy *query*, or a sequence thereof.

We have found little pedagogical literature about the usage of search tools: although it seems acknowledged that the usage of a search tool is a core part of self-regulated learning and probably other forms of web-based learning, we have found little studies that describe the most important values of a search tool in the learning process or even in the authoring process. What we describe below are expectations that we have formulated in the course of the many projects relating to ActiveMath as described in Chapter 9.

5.1.1 Access for Learners

In ActiveMath, the search tool is designed for the learner to achieve the following objectives:

- To discover or be reminded of essential statements, such as claims of important theorems or definitions: in this case the result should focus on conceptual elements with relevance to the current learning objectives. Moreover, the queries are expectedly small, quick, and approximate,
- To learn about the relationship among knowledge items, mathematical symbols, etc by the inspection of the items' relations starting at the view of an item.
- To discover new content items that can be useful in refining, enlarging or refreshing knowledge of the given concepts. Or to discover another example on a given topic - for example, a new exercise training mastery of a theorem.
- To communicate findings in electronic communications so that interesting perspectives listing available learning content can be shown to others.

Situations in which a learner may wish to see a content item outside of a book, or search for one, include the process of active learning exercising in which, for example, a rule, definition, or theorem has to be recalled and reapplied, in order to raise understanding. These situations also include the quest for more examples or more exercises relevant to a particular topic or containing a particular concept.

This access can be offered in many places in the form of a link; for example in the feedback of an exercise step, or as side-note in an example. With her own initiative, the learner could also use a search tool for textual search or formulæ search in order to recall and/or copy semantics to a particular application, e.g. a concept mapping tool.

As I have described in Chapter 2 the content of ActiveMath can be multilingual. Each item can be expressed in multiple languages. This means that users speaking several languages can see items in any language, changing from one to the other without losing their context. It also means that the search engine should handle multilingualism correctly: always search, first, in the user's language, but also search in other languages with less preference. In all cases, the language specific techniques of information retrieval, which include the stemming of words to obtain their radical (for example converting "spinning" to "spin"), need to be applied with care, each for their own language.

5.1.2 Access For Authors

Search can also serve authors and tutors, which will search content with textual, mathematical, and attribute queries in many situations:

- To find content items that could be re-used (e.g. about a given concept) by evaluating them rendered as a learner would see them: this will enable a decision to re-use, or to start writing anew.
- To find content to be assembled in a book using the assembly tool (see Chapter 2).
- To find content items to be adjusted or copied: in this case, the view of the item is followed by a call to the editor (the jE button described in Chapter 3)

In all these usage situations, the authors express their wishes using all possible queries: they may use words describing concepts, use formulæ appearing in the items, restrict by item-types or other characteristic of the item stored in its metadata*, or search for precise sequences of words.

5.2 Ingredients of a Search Tool

In the outline of the ingredients needed for a search tool I follow [You05] but our focus is on the mathematical content items and formulæ as opposed to the exclusive function orientation of [You05]. I present additional ingredients that are required for a system that does not only search but also presents the content.

Identifiers and References Content items need to be referenced in order to be displayed and linked to. References to items should be exchangeable in e-mail communications. Therefore, they need to be context independent, short, and, ideally, readable.

Storage, Extraction, and Presentation Mathematical content items are embedded within the larger context of documents. They need to be stored in repositories and easily extractable, so that they can be queried and presented.

The presentation of content items alone, as a dictionary does, could be offered; this is supplementary to classical navigation organizations such as books.

Because of the nature of the web, a user may also be viewing a content item for which almost no word or mathematical symbol is understandable to him. Supporting this presentation by links that can explain the concepts is an important role of a content-presentation architecture, be it within books or within the search display.

Query Input For a search tool to be efficient, information retrieval has shown, e.g. in [vR79] and [Hea09], that users expect multiple query-attempts with fast display of results; typically, these queries are refinements of each other.

Queries for textual fragments are well known: the simple input of words and possibly the degree of matching. This has been the focus of Information Retrieval, a classical domain of computer science.

Queries for item attributes can be input using form-based interfaces or using a dedicated syntax.

Queries for mathematical expressions is less developed. The usage of a facility to input the formulæ is important.

Indexing, Analysis, and Back-End Query Information Retrieval as introduced in [vR79] and others, long matured with refinements with probabilistic guarantees such as [TWS04], and made widely accessible in such software as Lucene [HG04] provides powerful methods to convert rows of tokens (e.g. words) to an index for which search results can be computed efficiently.

This relies on a tokenization process called *analysis* which, among others, *stems* words (e.g. removes plural), removes too frequent words, and performs other forms of language dependent transformations. Thanks to these techniques, the search tool execution is simplified to the execution of a combination of exact matching queries.

Information retrieval suggests that an index be built that stores, for each *term*, its occurrence in each document (this is called "document inversion"). This forms an index on which queries can be evaluated, and results can be returned that consist of a list of matches (pairs of a document and its score). The techniques allow the results list to be sorted by score.

Presentation of Results The way a search tool displays a result is very important because users often only visit the first few matches. Information retrieval uses the notion of relevance of a match: a score that is assigned to a document matching a query; results with highest relevance should be presented first. The sorting by relevance of results is fundamental for the management of resources: it provides a natural way to order the results and thus respond with a hopefully useful (partial) answer within the first page of results. Paging the results is fundamental for a reasonable response time. The relevance to the query may be the only criterion: the tf-idf formula described in [HG04] is often used for ranking: it is based on the number of occurrences of the term in the document (*the term frequency*) and the discriminating value of the term (*the inverse document frequency*). However several search engines compute weights to documents based on external information, for example the PageRank algorithm of the Google Web-robot*. The results' list should provide visual hints about the type of mathematical item presented, for example its title and its ranking.

In the learning process, the relevance is even more important as many of the concepts can be very foreign to the user, and thus they will have a hard time choosing between search results. Provided information can be obtained about the user's knowledge or objectives, a more specific ranking can be achieved.

Access to Content Presenting the actual content may or may not be a task for the search tool. In some cases, such as in general web-search engines, this is not possible, and access to the content can only occur via the provision of a link which, once clicked, forgets most of the search context the user was in.

In other cases, the search tool can present the content itself. This makes the tool close to a lexicon which can display extra information about the item or links to the contexts where such an item or concept occurs. Most server-embedded search tools are like this. The search tool, along with the presentation architecture and the other navigation methods, then becomes a complete browsing tool.

Access to the (appropriate) content item is the *ultimate* goal of the search process but, as described in [Hea09, Chap. 3], it is worth considering the search as part of a larger process of the user. Thus the sequence of queries of typical users of a search engine tend to be posed in successive refinements and to be influenced by the resulting documents they find. Also, the information seeking procedure may demand management: this can be solved by offering the *saving* of search queries so as to return back to them.

5.3 The ActiveMath Search Tool

The search tool of ActiveMath searches (an index produced from) the OMDoc sources. It provides a learner-friendly user-interface that combines plain-text search

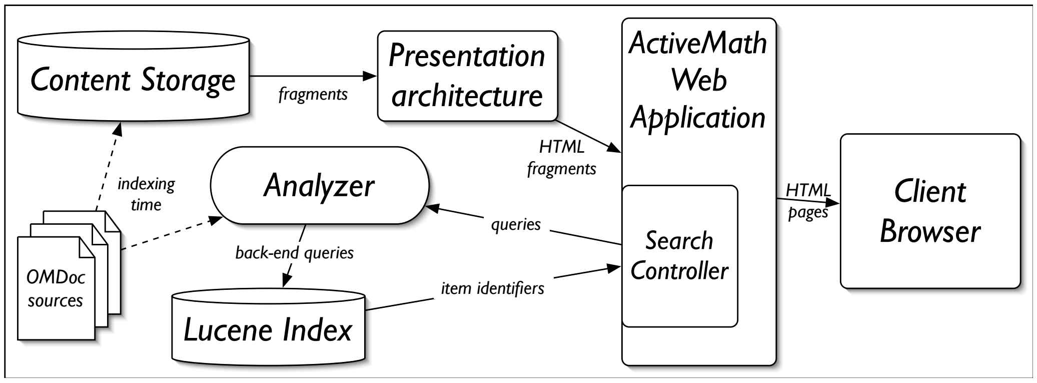
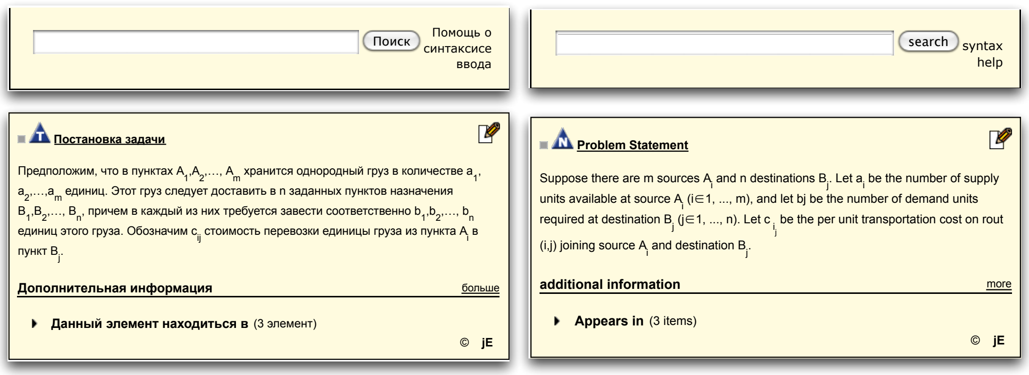


Figure 5.1: Architecture of the ActiveMath search tool



Постановка задачи

Предположим, что в пунктах A_1,A_2,…, A_m хранится однородный груз в количестве a_1, a_2,…,a_m единиц. Этот груз следует доставить в n заданных пунктов назначения B_1,B_2,…, B_n, причем в каждый из них требуется завести соответственно b_1,b_2,…, b_n единиц этого груза. Обозначим c_{ij} стоимость перевозки единицы груза из пункта A_i в пункт B_j.

Дополнительная информация

Данный элемент находиться в (3 элемент)

Problem Statement

Suppose there are m sources A_i and n destinations B_j. Let a_i be the number of supply units available at source A_i (i∈1, ..., m), and let bj be the number of demand units required at destination B_j (j∈1, ..., n). Let c_{i_j} be the per unit transportation cost on rout (i,j) joining source A_i and destination B_j.

additional information

Appears in (3 items)

Figure 5.2: The same item, hence URL, rendered by the search tool to different browsers.

with item-attributes and formulæ search. It presents the results of queries as well as individual items and their relations.

A coarse architecture of the tool is in Figure 5.1 which presents the flow of information at indexing time (on the left), at query time (right bottom), and at item-presentation time (right top).

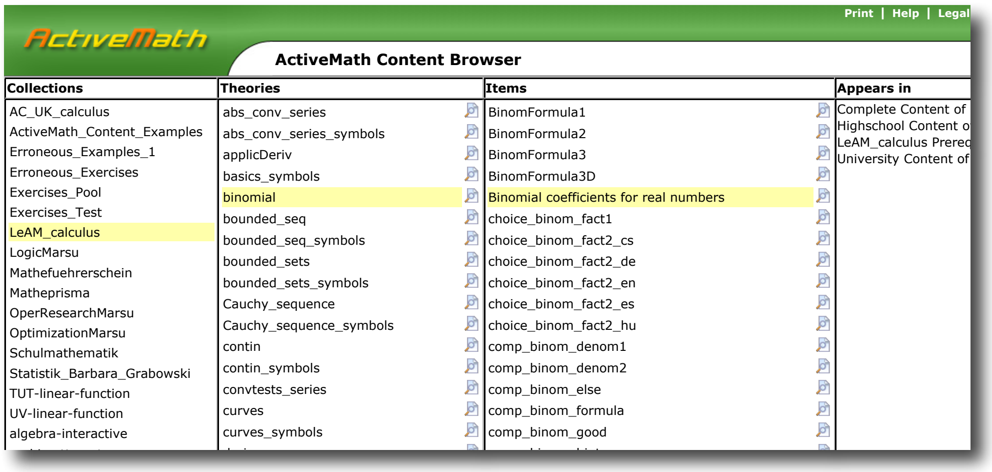
The description of the ActiveMath search tool follows the structure of the Section 5.2.

5.3.1 Identifiers and References

As explained in the storage description, Chapter 4, each content item is marked by an identifier expected to be unique per ActiveMath installation. References to content items are presented as links within the browser presentations which makes them exchangeable by most desktop applications. These references do not include all the user-information or the query, thus the same reference may point one user's browser to a presentation in English and another to a presentation in Russian; such a dichotomy is displayed in Figure 5.2.

The usage of URIs to encode references to items is the natural choice. In Active-Math, these URIs are HTTP* URLs that are clickable. They link to an *item view* whose URL is in the form:

```
/ActiveMath2/search/browse.cmd?id=mbase://collection/theory/name
```


Print | Help | Legal

ActiveMath Content Browser

Collections	Theories	Items	Appears in
AC_UK_calculus	abs_conv_series	BinomFormula1	Complete Content of
ActiveMath_Content_Examples	abs_conv_series_symbols	BinomFormula2	Highschool Content o
Erroneous_Examples_1	applicDeriv	BinomFormula3	LeAM_calculus Prereq
Erroneous_Exercises	basics_symbols	BinomFormula3D	University Content of
Exercises_Pool	binomial	Binomial coefficients for real numbers	
Exercises_Test	bounded_seq	choice_binom_fact1	
LeAM_calculus	bounded_seq_symbols	choice_binom_fact2_cs	
LogicMarsu	bounded_sets	choice_binom_fact2_de	
Mathefuehrerschein	bounded_sets_symbols	choice_binom_fact2_en	
Matheprisma	Cauchy_sequence	choice_binom_fact2_es	
OperResearchMarsu	Cauchy_sequence_symbols	choice_binom_fact2_hu	
OptimizationMarsu	contin	comp_binom_denom1	
Schulmathematik	contin_symbols	comp_binom_denom2	
Statistik_Barbara_Grabowski	convtests_series	comp_binom_else	
TUT-linear-function	curves	comp_binom_formula	
UV-linear-function	curves_symbols	comp_binom_good	
algebra-interactive			

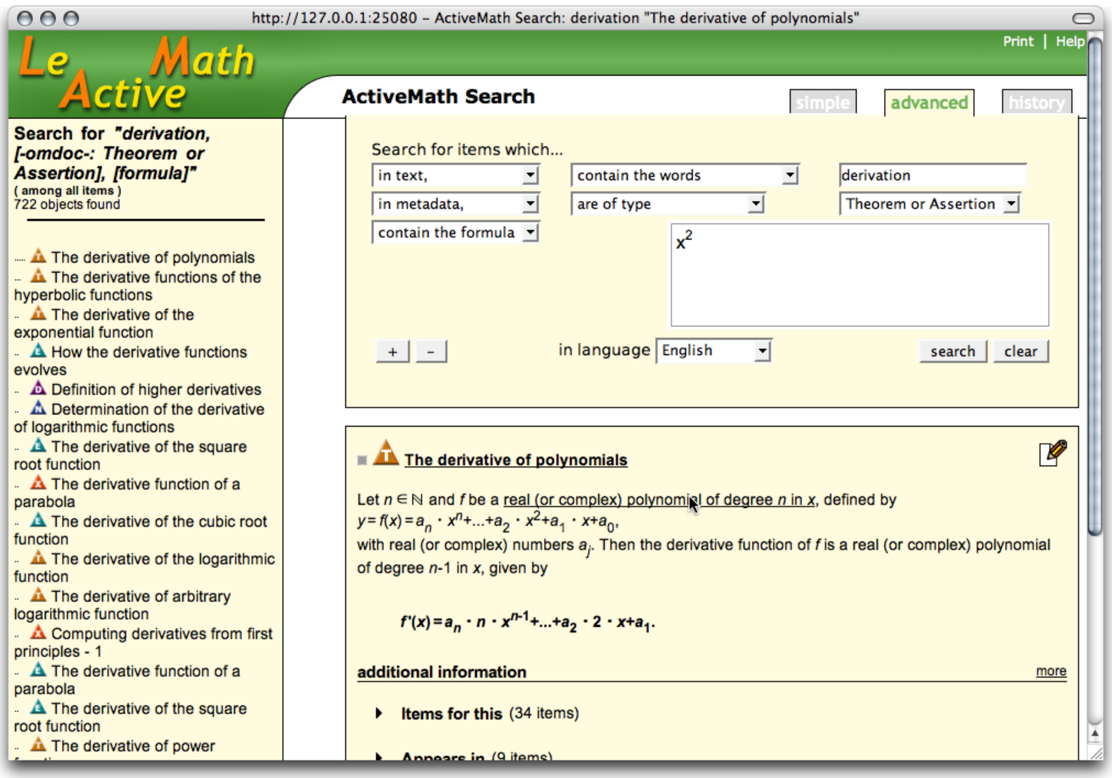
Figure 5.3: The browser view of a typical collection browsing that a Web-robot would be receiving after entering ActiveMath and further spidering along the links.

Such URLs can be exchanged throughout desktop operating systems, stored in electronic communication (emails, web pages, ...), or recorded in the browser history. Actions with such a URL are not limited to the simple *open* action: applications can analyze the URL and store it, or a derivative of it, as a reference to the content-item.

This is used by several tools around ActiveMath: the assembly tool [Hom06], the ICmap concept-mapping tool [MKH05], the dialogue console [DCF06], and the open-learner-model [NBM07] which all *speak about* items and offer a way to let learners see the item through the action of opening this URL (e.g. by a double-click).

URLs can also be dragged-and-dropped: ICMap, the assembly-tool, and the authoring editor jEditOQMath (Chapter 3) receive the drops of *item references* – the concept-mapping tool creates a node that represents this item while the authoring tool inserts a reference to the item (a link or a metadata* relation).

Since these URIs are also downloadable, they provide an entry point to a Web-robot*. Among others, I used this in a comparison between the Google and ActiveMath search engines described in the Section 5.4: the robots entering an ActiveMath server are presented with a page that browsers normally don't follow but which lists the content by collection, by theory, and by items, linking them to their item view and to the book pages where they appear (such a list is depicted in Figure 5.3).


http://127.0.0.1:25080 – ActiveMath Search: derivation "The derivative of polynomials"

Search for *"derivation, [-omdoc-: Theorem or Assertion], [formula]"*
(among all items)
722 objects found

- The derivative of polynomials
- The derivative functions of the hyperbolic functions
- The derivative of the exponential function
- How the derivative functions evolves
- Definition of higher derivatives
- Determination of the derivative of logarithmic functions
- The derivative of the square root function
- The derivative function of a parabola
- The derivative of the cubic root function
- The derivative of the logarithmic function
- The derivative of arbitrary logarithmic function
- Computing derivatives from first principles - 1
- The derivative function of a parabola
- The derivative of the square root function
- The derivative of power

Search for items which...

in text,	contain the words	derivation
in metadata,	are of type	Theorem or Assertion
contain the formula	x^2	

+ - in language English search clear

The derivative of polynomials

Let $n \in \mathbb{N}$ and f be a real (or complex) polynomial of degree n in x, defined by
$y = f(x) = a_n \cdot x^n + \ldots + a_2 \cdot x^2 + a_1 \cdot x + a_0$,
with real (or complex) numbers a_j. Then the derivative function of f is a real (or complex) polynomial of degree n-1 in x, given by

$$f'(x) = a_n \cdot n \cdot x^{n-1} + \ldots + a_2 \cdot 2 \cdot x + a_1.$$

additional information more

▸ **Items for this** (34 items)

▸ **Appears in** (9 items)

Figure 5.4: The advanced search user interface.

5.3.2 Storage and Extraction

The OMDOC content items' storage is described in Chapter 4: it processes OMDOC files in content-collections, makes them absolutized*, and stores them in separate content-items. At the time of storage reload, the search-index is delivered with a content-item to be indexed: its metadata* attributes and its content are run through the ActiveMath analyzer which produces several streams of tokens as described below.

5.3.3 Query Input

The user interface of the ActiveMath search tool allows for a set of queries to be *edited*. At search time, they are expanded to low-level queries, which refer to the index fields. ActiveMath offers several kinds of queries: text queries, attribute queries, and formulæ queries.

Simple queries are input in a text field. By default, they represent a tolerant text query of all the words. Power-users can use the query syntax which allows the input of a conjunction of text and attribute queries but does not yet allow mathematical formulae.

In the advanced search mode users edit a boolean combination of text, attribute, and formulæ queries: simple text fields are used for text queries, item-attributes' queries are input using pop-up-menus, while formulæ queries are input with the Wiris input-editor explained in Chapter 2. A screenshot of the advanced search user-interface with all query types is in Figure 5.4.

This *user-level* query is maintained in the history and kept in memory to return to it later using the history tab or using the link behind the summary of the search (the text starting with *Search for "derivation",* in the Figure 5.4). This query is converted into a back-end query following several policies that we expect match the users' needs:

- For learners, by default, the search is restricted to the conceptual content items of the book that was last read. A click allows to search through all item types of the complete platform.
- For learners, only some item types are presented. `symbol`, `symbolpresentation`, `omgroup`, and `interaction` elements are hidden. They are visible to authors (as registered in the user's profile).
- By default, the search, user-interface, and presentation of results, are done with the user's language (English in the case of Figure 5.4). Thus the proper *analysis* mechanism (which is described below) can be applied. However, by a recent change, other languages can also be searched for because it has been often observed that authored content inadvertently gets flagged to be in English but are in another language (for example, because the *xml:lang* attribute is missing and the default language is not changed in the DTD*). The precision clearly suffers because of such measures. As displayed in Figure 5.4, the language the items are being searched with preferably can be changed.

These filtering policies are applied at query expansion, when converting the user-level queries into back-end queries as described below.

5.3.4 Indexing, Analysis, and Back-end Queries

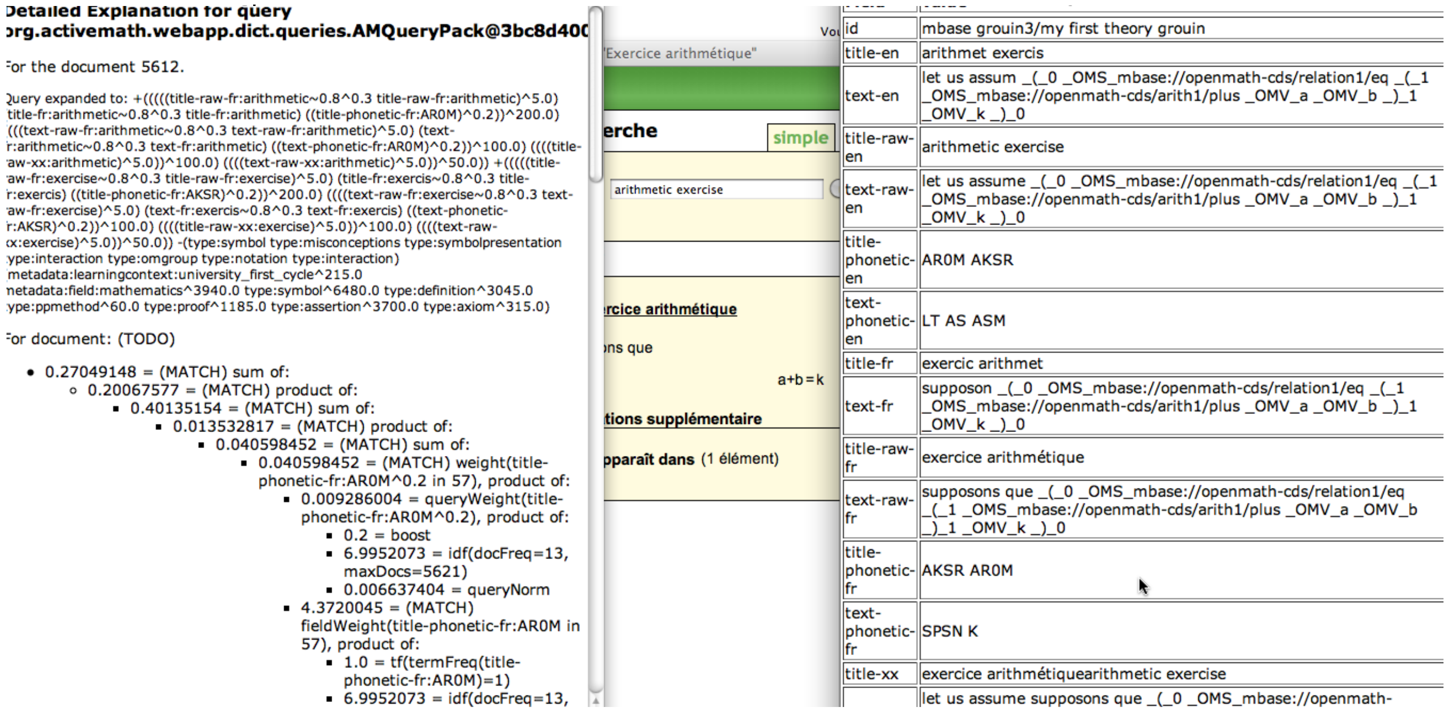
The indexing process follows the Information Retrieval approach: the content is read from the OMDoc sources and decomposed by the *analysis process* in parallel streams of tokens. These streams are indexed by the Lucene* library. Each token is stored on the disk along with its position in the stream in a way that allows queries for (rows of) tokens to be efficiently matched returning the number of occurrences and the document numbers.

I have developed an analysis process which converts the title, annotations metadata*, formal and textual content (including formulæ) to tokens. It allows all the queries above. The streams produced are as follows:

- For each metadata *property* with values in an enumarated set (see Section 2.2.2) a token is created (for example, the token `difficulty:hard` in the field `metadata`).
- For each language, three streams of tokens are created for the title's and body's text and formulæ: one is made of raw words (for language `la` the fields `title-raw-la` and `text-raw-la`), one is made of stemmed words according to the analyzer that we have found best for that language[1] (for language `fr` the fields `title-fr` and `text-fr`), and one is of the phonetic translations of words along the Double Metaphone* algorithm (for language `de` the fields `title-phonetic-de` and `text-phonetic-de`).
- A stream of words is created by merging all languages and without stemming (since the language is deemed unknown) (the fields `title-raw-xx` and `text-raw-xx`),
- The above streams of texts and titles also contain tokenization of the mathematical formulæ's OpenMath* elements following the breadth-first order. The tokens created are made of words for each operator, each bracket and its level, each number, and each variable occurrence.

So as to understand how the analysis is applied and how the matching is performed, two expert-oriented tools are available: the **analysis tool** is linked from an item-presentation when the user is logged-in as an author, besides the jE button described in Section 3.4.4. It provides a view of all the fields and token streams that the indexing process creates. The **search-match explain tool** is accessible by a double-click on the scores bullets that are shown on the left of each search result (for example, the first match in Figure 5.4 has five such bullets, on the left of the T triangle itself on the left of the title, *The derivative of polynomials*); the explain tool shows the low-level query resulting of the high-level query and, more importantly, the base-level queries which produced a non-zero match. This indicates, for example, the orthographic variation that has matched a fuzzy-query which allows typos. A screenshot of these expert functions is in Figure 5.5 where one sees that only the phonetic query has matched.

[1]See the source of `AMAnalyzer` for the choices of analyzers for each language. The rationale has generally been that the Porter stemmer [Por80], as provided by the Snowball packages of the Lucene* contributions provide a good first approximation but that dedicated analyzers may be better suited to handle other exceptions. Each new language requires experts to address this; the stemmers capabilities are verified in the unit tests `TestTokenStream`. Several tests there are commented out since the analyzer found did not perform enough.



```
Detailed Explanation for query
org.activemath.webapp.dict.queries.AMQueryPack@3bc8d400

For the document 5612.

Query expanded to: +(((((title-raw-fr:arithmetic~0.8^0.3 title-raw-fr:arithmetic)^5.0)
title-fr:arithmetic~0.8^0.3 title-fr:arithmetic) ((title-phonetic-fr:AR0M)^0.2))^200.0)
(((text-raw-fr:arithmetic~0.8^0.3 text-raw-fr:arithmetic)^5.0) (text-
r:arithmetic~0.8^0.3 text-fr:arithmetic) ((text-phonetic-fr:AR0M)^0.2))^100.0) ((((title-
aw-xx:arithmetic)^5.0))^100.0) ((((text-raw-xx:arithmetic)^5.0))^50.0)) +(((((title-
aw-fr:exercise~0.8^0.3 title-raw-fr:exercise)^5.0) (title-fr:exercis~0.8^0.3 title-
r:exercis) ((title-phonetic-fr:AKSR)^0.2))^200.0) ((((text-raw-fr:exercise~0.8^0.3 text-
aw-fr:exercise)^5.0) (text-fr:exercis~0.8^0.3 text-fr:exercis) ((text-phonetic-
r:AKSR)^0.2))^100.0) ((((title-raw-xx:exercise)^5.0))^100.0) ((((text-raw-
x:exercise)^5.0))^50.0)) -(type:symbol type:misconceptions type:symbolpresentation
ype:interaction type:omgroup type:notation type:interaction)
metadata:learningcontext:university_first_cycle^215.0
netadata:field:mathematics^3940.0 type:symbol^6480.0 type:definition^3045.0
ype:ppmethod^60.0 type:proof^1185.0 type:assertion^3700.0 type:axiom^315.0)

For document: (TODO)

• 0.27049148 = (MATCH) sum of:
    ○ 0.20067577 = (MATCH) product of:
        ▪ 0.40135154 = (MATCH) sum of:
            ▪ 0.013532817 = (MATCH) product of:
                ▪ 0.040598452 = (MATCH) sum of:
                    ▪ 0.040598452 = (MATCH) weight(title-
                      phonetic-fr:AR0M^0.2 in 57), product of:
                        ▪ 0.009286004 = queryWeight(title-
                          phonetic-fr:AR0M^0.2), product of:
                            ▪ 0.2 = boost
                            ▪ 6.9952073 = idf(docFreq=13,
                              maxDocs=5621)
                            ▪ 0.006637404 = queryNorm
                        ▪ 4.3720045 = (MATCH)
                          fieldWeight(title-phonetic-fr:AR0M in
                          57), product of:
                            ▪ 1.0 = tf(termFreq(title-
                              phonetic-fr:AR0M)=1)
                            ▪ 6.9952073 = idf(docFreq=13,
```

id	mbase grouin3/my first theory grouin
title-en	arithmet exercis
text-en	let us assum _(_0 _OMS_mbase://openmath-cds/relation1/eq _(_1 _OMS_mbase://openmath-cds/arith1/plus _OMV_a _OMV_b _)_1 _OMV_k _)_0
title-raw-en	arithmetic exercise
text-raw-en	let us assume _(_0 _OMS_mbase://openmath-cds/relation1/eq _(_1 _OMS_mbase://openmath-cds/arith1/plus _OMV_a _OMV_b _)_1 _OMV_k _)_0
title-phonetic-en	AR0M AKSR
text-phonetic-en	LT AS ASM
title-fr	exercic arithmet
text-fr	supposon _(_0 _OMS_mbase://openmath-cds/relation1/eq _(_1 _OMS_mbase://openmath-cds/arith1/plus _OMV_a _OMV_b _)_1 _OMV_k _)_0
title-raw-fr	exercice arithmétique
text-raw-fr	supposons que _(_0 _OMS_mbase://openmath-cds/relation1/eq _(_1 _OMS_mbase://openmath-cds/arith1/plus _OMV_a _OMV_b _)_1 _OMV_k _)_0
title-phonetic-fr	AKSR AR0M
text-phonetic-fr	SPSN K
title-xx	exercice arithmétiquearithmetic exercise
	let us assume supposons que _(_0 _OMS_mbase://openmath-

Figure 5.5: Expert search tools shown for the sample document of this chapter having searched the words *arithmetic exercise* with a user having French as main language.

5.3.5 Example Tokenization and Queries

Consider the following very simple content item in two languages (French in English complemented by a formula in the universal language:

`en`: **Arithmetic exercise**
`fr`: **Exercice arithmétique**
`en`: *Let us assume*
`fr`: *Supposons que*
`x-all`: $a + b = k$.

Our analysis process decomposes its content in named fields each populated with a sequence of tokens passed to the Lucene* library for indexing. For the content item above, the following tokens are provided to the index:

`id`	`mbase://collection/theory/arithExo`
`metadata:`	`type:exercise`
`title-raw-xx:`	`arithmetic exercise exercice arithmétique`
`text-raw-xx:`	`let us assume supposons que _(_0 _OMS_mbase://openmath-cds/relation1/eq _(_1 _OMS_mbase://openmath-cds/arith1/plus _OMV_a _OMV_b _)_1 _OMV_k _)_0`
`title-raw-en:`	`arithmetic exercise`
`title-en:`	`arithmet exercis`
`text-en:`	`let us assum _(_0 _OMS_mbase://openmath-cds/relation1/eq _(_1 _OMS_mbase://openmath-cds/arith1/plus _OMV_a _OMV_b _)_1 _OMV_k _)_0`
`text-raw-en:`	`let us assume_(_0 _OMS_mbase://openmath-cds/relation1/eq _(_1 _OMS_mbase://openmath-cds/arith1/plus _OMV_a _OMV_b _)_1 _OMV_k _)_0`
`title-phonetic-en`	`AROM AKSR`
`text-phonetic-en`	`LT AS ASM`
`title-raw-fr:`	`exercice arithmétique`
`title-fr:`	`exercic arithmet`
`text-fr:`	`supposon que_(_0 _OMS_mbase://openmath-cds/relation1/eq _(_1 _OMS_mbase://openmath-cds/arith1/plus _OMV_a _OMV_b _)_1 _OMV_k _)_0`
`text-raw-fr:`	`let us assume_(_0 _OMS_mbase://openmath-cds/relation1/eq _(_1 _OMS_mbase://openmath-cds/arith1/plus _OMV_a _OMV_b _)_1 _OMV_k _)_0`
`title-phonetic-fr`	`AKSR AROM`
`text-phonetic-fr`	`SPSN K`

With these token-streams in the index, queries for exact text, fuzzy text, item attributes, simple formulæ and formulæ with wild cards can be performed. They match the item each with a particular relevance score computed on the basis of the field and type of match (e.g., matches in titles are *boosted* by a factor of 5 as I expect them to be more relevant than matches in text):

- If the user enters "arithmetic" while working in English, the analysis converts this word to a query for token `arithmetic` in the field `title-raw-en` and for `arithmet` in the field `title-en` (and more). This is matched to our item yielding a high score.
- If the user enters "arithmetic" while working in French, the analysis converts it to a query for token `arithmetic` in field `title-fr`, for token `arithmetic`

in field `title-raw-xx` (and more). Only the latter matches which would match less than another item with that exact word in French if it existed.

- If the user enters "assuming", the analysis converts it to a query for the token "assum" in the *text-en* field and to a query for the token "ASMN" in the field `text-phonetic-en` (and other queries). These two queries are matching our item fairly.

- A query for the type exercise would be reformulated as an index-query for the token `type:exercise` in the field `metadata` which is matched to our item.

- If the user inputs the formula $a + b$ as formula query, it is translated to a query for the row of tokens
  `_(_`i `_OMS_mbase://openmath-cds/arith1/plus _OMV_a _OMV_b _)_`i
  where i ranges from 0 to the maximum-depth in the index. This is matched, with $i = 1$, to the tokens of our OpenMath* representation of $a+b$ and thus yields a fair score.

- the formula $? = k$ can be input as a query by assigning the wild card role to the $?$ sign. It is translated to a query for the token sequence
  `_(_`i `_OMS_mbase://openmath-cds/relation1/eq _? _OMV_k _)_`i
  where i ranges from 0 to the maximum-depth in the index and where `_?` is a wild card match of any row of tokens with the exception of the tokens `_(_`j with $j \geq i$. This can be exactly matched, with $i = 0$ to the OpenMath representation of our formula. The score of such a match, using the SpanNearQuery of Lucene*, depends on the length of the content matched by the wild-card.

5.3.6 Results Presentation

The ActiveMath search tool returns the first page of results. Each result is displayed with bullets indicating the score, an icon of its type, and its title. The user can click it to obtain a display of the item.

The plain-text search, by default, behaves first similarly to a book index. It only returns conceptual content items of the current book with a sufficient relevance; a click can generalize this query. The tool is complemented by links that trigger an equivalent query to external sources of mathematical content on the Web.

Clicking on an item in the result list presents the item view. The ActiveMath presentation architecture described in Chapter 2 converts the OMDOC source into a format that is highly readable and is linked to other functionalities of ActiveMath; for example, references to other items in the OMDOC source are transformed to HTML* anchors linked to its item view, exercises with interactive content are displayed with a link to start them.

Search results are presented with a short description which is a link that can be exchanged over the web. This allows users of the search engine to bookmark results, to communicate them by email or chat, or to insert in web pages, for example, as a first incentive within a learning-management system. Opening such a link will display the same view, with all search tool options in the originator's search view; this view is inserted as a new step in the history of searches for the user.

5.4 Evaluation of the Search Tool

The search tool of ActiveMath has been evaluated by three methodologies: The first is a formative user testing where acceptable performance is evaluated and the users are questioned about their opinions. The second is a typical search evaluation with measures for precision and recall. As a third evaluation, a comparison to the Google search engine has been performed.

5.4.1 First Performance Tests

The first tests were performed with the LEACTIVEMATH calculus content described in Section 9.3.2 – a corpus equivalent to a typeset book of about 500 pages in English with full translations to German and Spanish.

The index contained 2761 documents made of 560'259 tokens: 182'765 words and 377'494 mathematical tokens spread in 36'389 formulæ, and 13'681 attribute tokens. On disk, this index takes about 10% of the size of its OMDOC sources – about 10Mbytes. All queries are responded to, measuring browser time, in maximum 150 milliseconds.

Later indexes have been built keeping quite acceptable performance: with an index fifty times bigger than normal, queries (as above) take about 300 ms.

type	**query**	**evaluation**	**precision**	**recall**
plain-text	durchschnittliche	14 matches, 4 correct, no miss	0.286	1.0
plain-text	tangant	two results correct no miss	1.0	1.0
plain-text	sin	more than 20 matches, all wrong	0.0	0.0
formula	x^2	1 result correct, no miss	1.0	1.0
plain-text	theorem about quotient	1 correct match, no misses	1.0	1.0

Table 5.1: A few example queries, their precision and recall.

5.4.2 Formative evaluations

At the University of Edinburgh, in 2004, 12 mathematics students were invited to discover and use the ActiveMath learning environment under the supervision of an expert. The *think-aloud* protocol was applied. The evaluation indicated the learners found the ActiveMath search tool quite useful and enjoyed an acceptable ease of use. It suggested the importance of labeling content items by types and indicated that learners start grasping the structure of content items, when using the search tool. The evaluation revealed a few usability glitches, most importantly the incomprehensibility of the word *metadata* to qualify queries for items' attributes which was fixed later.

Three German high school classes have used the same tools and content for several weeks. First observations from log files indicate that 95% of the spontaneous usage of the search tool are simple text queries.

5.4.3 Precision and Recall Evaluation

One of the strategies to measure the quality of a search engine is to use indicator numbers that summarize the quality of search results compared to expectations. An evaluation for precision and recall is one such evaluation: it needs a test-suite of queries that can be input in the search tool, and an evaluation of each of the search results within the first page (the *good matches*), as well as a count of the missing search results.

The realization of such a test-suite should be done by an expert who knows the scope of the search well (what kind of queries should typically be used by learners) and that knows the content items that should be matching for each of the queries. Having this information allows one to compute, for a given query, the two following values after having first adjusted each value so that they are maximum 20, the number of possible matches shown in a page.

$$\text{precision} = \frac{\#\{\text{good matches found}\}}{\#\{\text{shown matches}\}} \text{ and recall} = \frac{\#\{\text{good matches found}\}}{\#\{\text{expected matches}\}}$$

Such measures are classical – a deeper explanation with a proposal of several other indicators of retrieval quality is in [Mah06a].

The first sample was created and measured by me against expected matches in the pre-recorded book *Up and Down* of LeAM_calculus*. A few queries with their precision and recall measures are displayed in Table 5.1, the complete list being in [Lib06].

The mean recall value obtained 0.93 (very high). The mean precision is 0.63, which is low since the fuzzy matching uses both the phonetic and edit-distance[2] matching approaches.

Since it requires a priori knowledge of the expected matches, this sample of 40 queries in a book of 30 pages is small. However, for mathematical material, there seems to be no classical sample collection available such as the massive ones gathered for the TREC competitions.[3]

Further precision-and-recall evaluations have been pursued in the Math-Bridge project in 2010. The first rounds, within the `Schulmathematik` collection[4] of test have led to a mean precision of 0.12 and recall of 0.81 whereas the same queries have led, after a first adjustment to the weighting to a mean precision of 0.53 and recall of 0.85. The test-suite was created by a physics didactics advanced student of Saarbrücken.

The same mission was given to all partners of the Math-Bridge project. To date the following means have been attained:

- for Spanish, a test-suite for the collection LeAM_calculus* has been realized yielding a mean precision of 0.79 and recall of 0.87. It was assembled by a mathematics lecturer in Madrid.
- for German, a math-didactics expert created a suite of query for the book of ActiveMath content that Math-Bridge has been transcoding (from TeX* to OQMath). The resulting test suite had 78% of mean precision and 84% of mean recall over a sample of 37 queries among 200 items.

5.4.4 Google Comparison

Another approach to evaluate a search engine is to compare the engine with another one which can also retrieve the content. Since ActiveMath is on the Web, it can be visited by a Web-robot*. I used the ability of the Google search engine to restrict its search on a given Web-server to compare Google results to the one of the ActiveMath search tool.

In 2006, I gathered another set of queries, expected to be matched in the complete content of the collection realized in LEACTIVEMATH [LG06] and compared

[2]The Lucene library offers a form of fuzzy queries which applies elementary modifications to the query words and recompute the matches. They are returned with a lower score based on the *edit-distance*, the amount of elementary modifications applied. Fuzzy textual matches in the ActiveMath search tool also use these queries.

[3]The TREC competition is a yearly competition organized along the TREC conferences by NIST where large collections of texts are given to participants, followed by queries. The result is evaluated, among others for precision and recall, by NIST. See `http://trec.nist.gov/` and the description in [Mah06a].

[4]Please refer to *list of contents* of the ActiveMath website where, among others, the `Schulmathematik`, originally authored by Edgar Kessler is described.

query	ActiveMath amount matches	Google amount matches
whenever function differential quotient	1	1
tangent convex concave	15	8
parabola maximum	6	17
water maxmimum	39	0
tangant maximum	31	0
sin maximum	48	0
inflection point	243	27
inflection point (no fuzzy)	20	27
sketch	69	1
cauchy sequence	20	65

Table 5.2: A few example queries with the number of matches in ActiveMath and Google search tools.

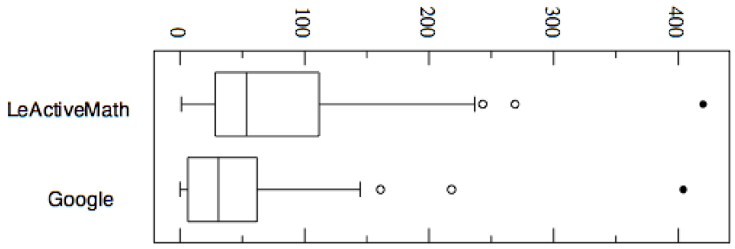


Figure 5.6: The distribution graph of both ActiveMath and Google matches

the number of matches. Selected results can be found in Table 5.2 and in full in [Lib06]. The Student-T-test comparison between the two columns indicated a t-score of 5.41 which indicates a significant difference. Distribution graphs of the number of matches are depicted in Figure 5.6 which indicate a broader variation of the ActiveMath search tool. One of the main differences of the ActiveMath search tool compared to Google is the number of matches (good and bad) introduced by the fuzzy matches which the Google search engine does not provide. Two other factors led to the differences: the dates of the index construction differ; and the fact that Google searches the text of the HTML* item views which means, for example, that the text of the relations from the items, which are shown along the presented item, are considered to be part of the item or that the words of a presented formula, such as the word *sin* in $\sin x$, are matched as well.

5.5 Related Work

A few search tools allow for the query of mathematical terms, e.g. that of HEΛM [AGC$^+$04] or MoMM [Urb04].

Because of the lack of a formal library, the ActiveMath search tool cannot manipulate the formulæ applying formal knowledge, for example term-ordering normalizations or symbol generalization. This results in a relatively low tolerance in formulæ search. In comparison, the ActiveMath search tool is intended for a broader learners' audience and benefits from a highly tuned user-interface.

The search tool of the Digital Library of Mathematical Functions explained in [You05] is dedicated to functions. As a result, it performs several normalizations of mathematical terms such as the conversion:

$$\frac{a}{b}\cdot\frac{c}{d} \text{ in } \frac{a\cdot c}{b\cdot d} \quad \text{or} \quad J\cdot H \text{ to } H\cdot J$$

(the details of these normalizations are in [AY07]).

Such normalizations introduce tolerance within the search tool and thus provide a higher recall. The usage of a simple type system for OpenMath* objects could enable the ActiveMath search tool to perform some of these normalizations, but it remains unclear whether the likelihood tto leave the user in confusion will be kept low: both transformations of the normalizations above are explicit cognitive processes in some learning processes (when learning to operate on fractions, around the age of 12 or 13 in France and Germany; when discovering commutativity around the age of 18 in these countries). When the process is explicit, the normalization must be avoided as it would otherwise let the learner believe that such a transformation is trivial. In most other cases, such transformations are part of the background reasoning of the users and are likely to be expected.

The ActiveMath search tool also differentiates itself from the DLMF search tool by the user interface: while the DLMF search tool defines a plain-text input syntax resembling the widespread TEX syntax, queries in the ActiveMath search tool are realized by input of concrete words, attribute-value, or formulæ.

Another avenue has been explored by Paul Cairns in [Cai04] where Latent Semantic Analysis can be used to provide a *semantic* distance between token-vectors, including mathematical terms. This additional fuzziness has been implemented for ActiveMath (see [Jed09]) but is not enabled by default since it may raise the imprecision.

MathWebSearch [SK06] is a Web-robot* for documents with MathML*-content. This tool uses the term indexing techniques of [Gra96] to index formulæ collected on the Web. Compared to this search engine, the ActiveMath search tool is more learner-oriented but still lacks the ability to perform queries for formulæ with variables that occur several times and would be replaced by arbitrary terms (joined wildcard queries). Since it uses the Lucene* library, the ActiveMath search tool appears better scalable compared to the term indexing technique which needs an in-memory representation. More importantly, the ActiveMath search tool presents the search results from the best match on. This allows the learner to focus on the first few matches in many cases to obtain the desired content item. The MathWeb-Search tool, however, seems to have no way to rank results and, indeed, orders arbitrarily a search result whose list is too big to fit on a page.

From the overall access point-of-view, the ActiveMath learning environment seems to be one of the rare mathematical content item's presentation servers which manages fine-grained items. A few similar projects are the Thesaurus at `http://thesaurus.maths.org/`, whose goal is an international dictionary of mathematical concepts, or the encyclopedia projects, such as WikiPedia (`http://wikipedia.org/`), PlanetMath (`http//:planetmath.org/`), or MathWorld (`http://www.mathworld.com/`). ActiveMath is the only one which offers search by attributes and formulæ of the semantic nature of the content encoding.

5.6 Conclusion

In this chapter, I have described the access to mathematical content items in ActiveMath through its search tool and presented evaluations of this search tool, which indicate positive results.

The main contribution of this research is the development of an analysis process for mathematical formulæ which converts them to streams of tokens. The information retrieval techniques can be applied as well as the joined contributions of tools of this domain adjusted for the purposes of mathematics learning.

Using the Lucene* library for indexing makes the search tool efficient and thus enjoyable to use and explore with.

The search interface has been designed for learners, and it has been evaluated for usability. Two main conclusions can be drawn from the evaluations:

- Not surprisingly, fuzziness introduces noise into the search results. Finding better ways to indicate the relevance is still desired.
- Conversely, the fuzziness introduces tolerance to the queries which is an important feature in our application. Fuzziness in formula search has not been explored yet. As explained above, normalization would be a first way and could be explored provided it is possible to indicate when a normalization is desirable or not for a learner.

Chapter 6

Publication and Re-use of Content for Web Platforms

The production of E-Learning content is known to be expensive, but its potential for reproduction is much greater than paper-published content. Thus, the idea of re-use of content has emerged and been studied, for example, in [RCM06], [BDFI06], and [MHRS06]. We have found, however, that very few studies address the management of long-term content evolution together with the actions of re-use such as aggregation, transmission, and publication.

This chapter advocates the notion of a **content-collection** corresponding to the organization of content projects in **shared directories**, along with the mechanism of **item-inclusion**, and the practice of **semantic content**. Together, these principles allow for project-based maintenance, connections to author communities, and the realization of specific learning experiences where re-used content appears as a coherent entity.

This chapter attempts to explore sharing and re-use methods, taking the distributed nature of these activities in account. The Web, that is the set of web-servers which are globally accessible, is the best means for this distribution.

We propose a model of sharing which supports' the **long term re-use and quality development of content**. It describes how this model is implemented in the ActiveMath learning environment platform and the supporting infrastructure we experimented with.

It starts with an an ideal re-use scenario that seems realistic according to the study of working teachers described in [GT08a], proposes a few definitions, and surveys current practices of re-use. A survey of the publication spaces of content and the various relationships an author can have to these spaces in the re-use paradigm is then presented followed by a description of recommended best practice. We conclude with a review of related work and an outlook to open questions.

This chapter is a revision and extension of [Lib08].

6.1 A User-Story of Re-use

Let us imagine an author who assembles the content for next year's course. For this purpose, a new content-collection is set up. It starts as an inclusion of last year's project but the author wishes to add more content and does not wish to affect last year's content. So his collection *extends* last year's collection by referencing it.

The author also wishes to include some interactive exercises that his long-distance colleague shared with him when they last met at a workshop, as well as the high-quality real-world examples of a big industrial project he encountered in his professional associations' online community. To be able to evaluate the content elements, he needs to see web-pages that describe the content projects, that point to public demos and that provide the conditions for re-use (which can be a sales action or an open-content license). These information pages allow him to track the online communities where these content projects are discussed and where their inclusion is documented.

Using the simple *import* facility of his authoring learning-platform (which we define in Section 6.2), he can see the *books* of these collections and can browse the content within the realm of his own server. When he starts to assemble the content of the first lessons, he quickly realizes that his students will need more technical instructions for his colleague's interactive exercises, and also that some of the real-world examples refer to concepts that are formulated differently than what he wishes to convey.

These adjustments are possible for him since he knows how to change the sources of the documents that are copied and processed by his authoring tool. Doing this, he creates an *appropriated* version of each of the three collections which are the basis of the course he is preparing. The content project of this year is first built with just a few *books* that imitate the books of last year and slowly get adapted to incorporate the two new content projects and further explanations.

After these modifications, the author can upload his course to the school server. This means uploading all four collections to this server (three base collections and the collection of the new course). Students will see it as a coherent single course. Advanced users of the school's server (e.g. remote teachers that see the public preview), will still be able to click through it down to the original collections, following an information page with *copyright information*.

Figure 6.1 presents the view of the author, in the central rectangle, with imported collections linked to their external *repositories*, and with a link to the target learning environment.

Because his way of working has clearly identified the derivative nature and the origin of each content collections that he is using, our author can incorporate in the new course, a few months later, corrections to last year's course or enhancements to the industrial examples. Similarly, he is able to transmit the enriched technical instructions he wrote so that his colleague considers them for inclusion in his repository...

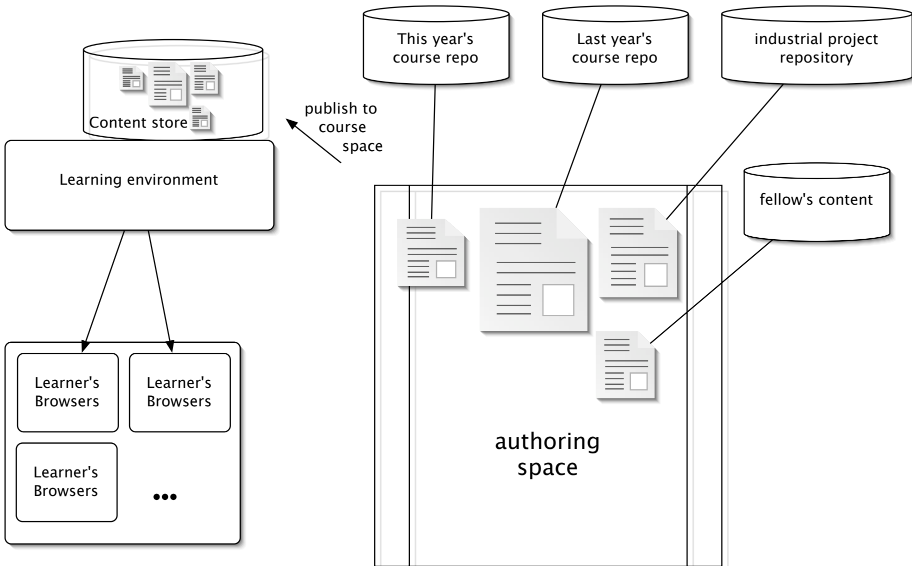


Figure 6.1: Organization of the author creation process as described in the user-story of Section 6.1.

The remainder of this chapter explains how such a user-story can be realized and presents the tool-set applied for authors of the ActiveMath learning environment to this end.

6.2 Elements of Re-use

Re-use simply means the action of taking an *existing* piece of content to use in one's own creation. The easy copying mechanism provided by computer processing combined with the global availability of content makes re-use appear as an interesting solution to the high cost of novel content creation. However, so far, it has happened rather rarely.

As described in Chapter 1, the authoring activity is within the context of several spaces, such as the composition space (world of the editor) and the preview space (a small learning platform which we call the authoring server) and it may lead to a publication space (for other authors) or a staging space (for the learners).

Re-use sheds a new light on these spaces since content projects are distributed across several instances of these spaces; therefore communication between these spaces is needed. Communication typically runs from the original author's publication space to the composition, staging, and publication spaces of the recipient authors. This chapter articulates re-use under the perspective of the many relationships between authors and these spaces which are mostly all visible on the Web.

Central to the re-use actions is the **granularity**, the level of detail to which the content items can be manipulated. The finer it is, the more management actions may be required but the more freedom for re-organization it allows; the bigger it is, the easier the content items are to exchange, but the the risk of requiring a new version to be created is higher.

As explained in Chapter 2, ActiveMath has chosen a very fine granularity where individual mathematical content items such as a definition, a motivation, or an example, are the content items. Most other tools have the granularity of a page (typical of content-management-systems or wikis), or of a book (typical of a desk-top document). Items in ActiveMath are combined into larger pages or books using the item-inclusion paradigm, that is, items are *assembled* (or *aggregated*) in to table-of-contents hierarchies forming books which can be browsed. This approach is what we call **item-inclusion**.

Another important aspect of re-use is the nature of the material and its content-encoding. The nature of the material varies from simple pictures to complete web-sites, and the nature of the content-encoding varies from final-delivery executable materials (such as a multimedia CD or a video exploration), to the full authoring sources. Each of these dimensions impact strongly on the re-use possibilities. For example, lack of authoring sources (or available software to edit them), means that acceptance or rejection are the only possibilities of re-use.

Because web-delivery technologies are changing rapidly, it is common for the authoring sources to be more abstract. One way to abstract is to write **content semantically**. For ActiveMath this has meant that all formulæ are encoded using the OpenMath standard and that all content-items are given a mathematical role (e.g. exercise, definition...). This has the advantage that mathematical notations can be homogenous within an authoring project (see Chapter 2).

Re-using a piece of e-learning content within another setting is a fundamental action to evaluate its facets. If the origins are sufficiently traced, it is possible to stimulate the evolution of the content pieces following this re-use and thus enable long-term **quality management**. Several such workflows which combine quality assessment and content composition have been studied in the e-Quality project (see, for example, [Mon05]).

Last but not least, the **consistency** of the resulting learning experience is important for learners: one of the justifications is the *coherence principle* of [CM02]. This fundamental property of the re-use result has to be ensured by the re-using author. Having triggered the re-use of a content collection, the set of relations and metadata of the platform (see Chapter 2) is affected: the set of items available to the tutorial component, user-model, or search engine is updated. Depending on the topology of links, this may yield new behaviours and new results which might or might not be of help to learners. In many cases, a re-using author may have wished this effect – for example the provision of new examples for existing concepts.

6.3 Current Practice in Re-using Learning Content

In this section we attempt to classify the current practices that can be described as re-use in e-learning. Alongside, we catalogue the weaknesses and strengths of each.

6.3.1 Classical Re-Use: Verbatim Inclusion

The simplest form of re-use is achieved by importing a file created by someone else, not changing it, and including it within one's own authoring space. This practice is common, for example, for the re-use of pictures from clip-arts libraries or in tightly-linked design teams which are organised in workflows as described in [CLM$^+$08]. Verbatim inclusion does not work as well as soon as changes are needed: for example when some wording needs changing, or some contextualization needs to be provided. Moreover, verbatim inclusion, e.g. using a file found on the web, may run the risk of loosing track of its origin: exchanges about further quality enhancements would not be possible.

6.3.2 Classical Re-Use: Copy-and-Paste

Most of the time, re-use is based on the usage of the *copy-and-paste* paradigm. Often, this is even the only method; for example, the evaluation of the WINDS authoring tools [KSO04] states: *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.*

Re-use through copy-and-paste is of great help for imitation, which is important for learning authors. This learning effect for authors is strongest if the fragments to be copied are presented in a form that explains how to re-input them; a simple example is the plain-text-sources as described in Chapter 3. However the copied piece may also become incorrect when inserted, as it has lost its context, which may include important information (see Chapter 4).

Re-use based on copy-and-paste and subsequent modification is the *quick and dirty* method that may suffice for many applications, but that is clearly insufficient for quality development as it loses the link to the origin and the scope of the inclusion (from where to where in the re-authored work). It thus prevents transmission of further enhancements except if the re-using author is able to remember and re-express it all in a later communication.

6.3.3 Classical Re-Use: Linking

The simplest form of re-use in web-publishing is realized by the usage of a hyperlink from the published text into another.

Linking is a method that is very easy to apply by authors and that keeps track of the origin (hence can take advantage of enhanced versions). However, when used by learners later on, it loses almost fully the users' context within the learning environment (which includes the advanced services of e-learning platforms such as user-modelling, personal course management, integrated tools...). It also deprives the learner from the context and prevents teachers from making modifications needed for their own students. Linking relies on servers outside the control of the e-learning providers, which may become an issue.[1]

Therefore, advanced forms of links make their appearance in the market. The emerging IMS Learning Tools Interoperability (IMS-LTI*) specification is proposing a form of enriched link which allows a learning-management-system to safely direct a learner's browser to a learning-tool, having shared his or her identity.

6.3.4 Classical Re-Use: Copy and Branch of Large Bodies

Another common practice is *branching*: this operation is typically achieved when moving from one project to another by simply duplicating the document and arranging the various parts. This practice is common in many versioning systems. It is also often performed by duplicating large documents (e.g. a complete course document) when starting to author their content.

The branching practice could be best of breed in principle but, in general, needs to be combined with further edits: for example, re-using a textbook chapter or the slides of a whole lesson will very likely lead to modifications.

Analysis of differences so as to allow enhancements to flow back and/or forth, as described in the user-story above, may become difficult with such a practice because of how detailed its editing of content items is. Indeed the usage of a differencing tool is then required and is rare because of its intrinsic difficulty.

[1]This fear so common in the USA that a senator proposed to ban federally subsidised schools from using many sites including Wikipedia [Gra07]

6.3.5 Classical Re-use: Channel-based as in News-Feeds

One of the common practices of web-based re-use, though not of e-learning content, is the paradigm of newsfeed-aggregation: within the publication of pages of news, it is possible to *aggregate* news from other sources and merge them in one news-page. The resulting stream of news includes both locally created news as well as remotely fetched news. They are generally presented in a chronological manner and serve targets as web-logs or groups' news pages. This paradigm is most commonly exercised by the subscription to RSS-channels, which are simple URLs of RSS or Atom documents, two XML formats which describe a timed sequence of texts.

Channel based re-use is very easy to activate: one only needs to drop the URL into the configuration of the aggregator.[2] The RSS documents can be easily cached, which yields resistance to network breaks and performance issues as long as the news-feeds stay within a moderate size.

Channel based re-use is not appropriate for e-learning: the same unpredictability as linked re-use is there, the time-oriented nature seems inappropriate for learning material and finally modifying an included news item is not a normal practice (so much that many tools often do not refresh their entries once downloaded).

6.4 Project- and Inclusion-based Re-use

In the previous section, we surveyed widespread re-use practices and saw their limits. In this section, we propose a model to content sharing and re-use for e-learning that appears to combine the advantages of each of the methods described above. To our knowledge, this model is new although it seems to be closely implementable in Connexions* (but see Section 6.10.2).

We propose to organize re-use along ***content projects***, which we call *content collections* in ActiveMath. Often they are created right after the planning stage, when a content goal is formulated and have a lifetime as long as the evolution of the intended learning experience (which may span several years spent at teaching a similar lecture or at maintaining a valuable set of resources). Content projects have a broader scope than single books or single lectures as, for example, they may gather several cooperating authors which agree on common work for a given theme without each of them being forced to use all of it.

In order to realize the re-use action of content projects, the learning environment needs to provide the ability to *import a content project*, i.e. the actions that bring the content projects within the scope of includable content in the intended learning experience. Such an action does not mean the inclusion of the whole content within the current authored work but only the enrichment of the accessible content. This enlarges the platforms' overall set of items, thus offering more choices

[2]An example of an aggregator is the Planet Feed Reader, `http://www.planetplanet.org/`.

to, for example, the search engine or to the tutorial component. Further, we propose that learning environments allow imported content to be ***included*** within the realization of another content collection, that is: its content-elements become part of a larger entity, for example content items become part of a *book*, a navigation artifact; or a notation to become part of the rendering process. Other actions with content projects include the browsing evaluation, the download, the appropriation, and the publication, all of which are tasks of an authoring support tool.

ActiveMath's content collections implement this pattern: to the eyes of the ActiveMath learning environment, all content elements are contained in a collection and are pulled from one common interface as described in Chapter 4. The collections are directories that contain a descriptor, some server-content-files (OMDoc files) as well as static web-resources (see Section 4.4.1).

Content collections should, at best, be thematic. They should have a long lifetime and carry a community of authors and consumers interested in the collection; this stands in sharp contrast to single documents shared, for example, in learning-objects'-repositories such as MERLOT [McM04]. Within such a long lifetime, quality and evolution can be managed and different views can be exchanged about the usage of the content. As a result, content-collections may come with their own versioning repository, their own download space, their own web-pages and their own community space.

Designating a content project as the **contents of a shared directory** without the need to move them around, thanks to the inclusion mechanism, allows the proper management of content-items while keeping a common file organization shared between contributors of the project. Thus, the recipient of a re-use action is able to identify whether what he changes can be replicated by other people, if there is interest. This location tracking is precisely what is missing in the copy-and-paste or the copy-and-branch re-use which both lead to an unmanaged proliferation of (mild) duplicates.

If the content of the shared directory is maintained with the help of a versioning server, the author has the possibility to receive and view updates from other authors within his local copy of the directory and try them by previewing them. Similarly, he is still able to characterize the changes compared to the shared directory of the versioning server. Thus the flow of further updates can be managed.

Advantages of the linking paradigm of re-use remain in a learning environment that allows reference-based inclusion. Indeed, the inclusion within a book's table-of-contents is a single line in ActiveMath's OMDoc. This inclusion is stronger than linking since it really inserts the content-item within the scope of the other items. Moreover the fine granularity as well as the semantic nature of content helps considerably in this re-organization since it allows, even at the paragraph level, re-use and modification as well as consistent math-notations even though the source may come from a totally different origin.

6.5 Re-use Along the Content Spaces

Having defined the notion of a content project, we now describe the re-use actions for the *spaces of content*.

Recall from Chapter 1 that we call *content space* any entity where content is considered to be *situated*, that is, where any user would find the content. This includes the storage of very different encoding, and with different functions (read, read-about, write, update,...), and different roles. The definitions of Chapter 1 has defined the following authoring spaces: authoring sources, platform server storage, and shared content storage.

6.5.1 Re-use and Authoring Sources

The authoring sources are the data of the tools that are manipulated by the authors to input and modify content. A regular authoring workflow associates particular authoring sources formats and tools to particular content types. In the re-use world, authoring sources play the important roles of being most probably the best data to edit in order to create modifications of the content.

Editing the sources of the re-used content seems to be the only method for the recipient author to transmit his modifications back to the original author or community. Therefore, it is important that authors share the sources within an open content project.

Conversely, authoring sources may not be shared by professional authors who only wish to distribute the finished product. This is typically the case of editing companies for which the organization of the sources is closely bound the team organization, associating, for example, the content-fragment types with the role that can best produce them. This is the reason a dedicated repurposing tool has been described in [RZM$^+$08, MHRS06] aiming at modifications of professionally created learning resources in their delivered format, HTML*.

6.5.2 Servers' Content Storage

- In order for it to be used in the learning environment, the content enters a storage mechanism which allows the environment to perform its (intelligent) services efficiently. We have described the content storage of ActiveMath in Chapter 4. It has three points of access for re-use to happen:

6.5.3 Published Content Storage

Authoring sources that are stored in files can be published over a *file-sharing interface* such as shared file-servers as well as WebDAV and versioning servers. Shared file servers work well in very small workgroups where it is clear who works on what. As soon as there is a risk of discoordination however the usage of a versioning server is a must that few groups avoid: it provides the safety net of reverting to earlier versions and documented and discretized commit operations.

Packs of authoring sources stored in files can also be published in archives. This common practice is widely used and works well for release-oriented publications. This applies to far-away-recipients such as publishing houses' customers, but is inappropriate for tight collaboration with frequent updates or where the complete editing process happens on the web-platform. Archive packages canbe, for example, the SCORM* packages. The practice of sharing archives yields a high risk of loosing track of the origin and it makes it harder to contribute modified content back to the originating authors.

Other mechanisms for accessing the source of the content exist, such as the interfaces on the back of MediaWiki of Confluence* [Atl06]. But they seem to be performing a very similar task to versioning servers and are much more proprietary.

6.6 A First Approach: a Shared Authoring Server

It has been suggested that a single global server would allow the re-use scenario of Section 6.1 to be easily realized. In this section we explain, beyond the performance issues, why central servers such as Wikipedia or Connexions* are not sufficient for re-use of learning-content.

The first impediment to the usage of a single server is the remote characteristics which attracts mistrust of educators who wish to be sure that their teaching will not be bitten by a network cut or by an unexpected change from a third-party.

The second impediment is much deeper: it is the scope of a content change. There is no reason that a single content author might change, for example, a WikiPedia page for the purposes of his intended teaching experience; wikipedia pages are meant for the general population and not for a single classroom.[3]

In the ActiveMath learning environment, for example, the addition of an exercise anywhere makes it a candidate for selection by its tutorial component [Ull07], or displayable by the search tool. The requirement to manage appropriate *scopes* to restrict impacts of changes appears and is provided by the separation between the many platforms.

[3]Actually, the usage of WikiPedia for education is common: the scenario can be to simply use the official WikiPedia or to copy the content to somewhere else and modify it there, an example includes `http://schools-wikipedia.org/`.

A global server, however, is clearly interesting as a content commons for the instructors and authors who wish to discover and share content items. The shared platform allows easy cross-linking and easy searching – two important facets for the re-use of new materials. However such a shared platform does not live very well with several branching versions of the same content (which we could call *mild duplicates*).

This shared platform is responsible for the success of the Connexions* project [HB04]. Connexions, however, does not aim at a rich adaptive experience to individual learners but helps the author realize a document that he can deliver to the learners within a different environment.

Most probably, domain-oriented author communities (Connexions* usage has been analyzed in [PNKJ08]) are the best hosts forsuch content-commons but the need to *export* from a content collection on a content commons to one's own learning environment remains.

6.7 Freedom to Re-Use

Re-use relies on the simple fact that authors provide the allowance that their content be shared. Many authors still forget such an allowance, neglect it, or are accepting of the fact that it is restricted.

Solutions for an author to give this freedom, that is to *license for re-use*,[4] are multiple. The most significant are provided by the multinational set of licenses of the Creative Commons intiative [Cre08]. Guiding documents to stimulate open-licenses for e-learning content have also been realized, e.g. [Kre07].

An important choice for an author to make when applying a license is that of allowing re-use with *derivation*, that is, with the right to republish in a modified form. Even though there is often some reluctance to allow this right, it is clear that it is fundamental. To our experience, the adaptations thus allowed are extremely common.

6.8 How can an Author Re-use Content?

Having described the spaces where the content projects and their sources can be stored, we now try to concretize the user story of Section 6.1.

The relationships an author has to the content storage spaces of a content reach from the loose browser connection with a remote server, all the way to a set of files that the author edits on his own computer. Content projects can be stored more or less remotely by the author and it may be possible to change this distance by transporting the content from one place to another in order to re-use it. Players

[4]The verb license comes from the latin *licere* which means *to allow*

in these transactions include the authoring tool, the user interface for the author, various preview servers (from local ones to broad showcase servers), community spaces, and file-sharing mechanisms. With a more detailed description later, the possible relations an author can have to a content storage space are:

- the author may be remotely browsing a collection that he can only read from,
- he can have his own learning platform connected through a web-service channel to a source of the content,
- he can have this entire content collection loaded in his learning platform, using a download of an archive or using a checkout,
- he may be using an *appropriated* version, that is a locally modified version of another collection.

Each of these connection relationships entails different possibilities for action which we describe below.

6.8.1 Remote browsing

For remote browsing the author's web-browser merely has to be a client of a learning platform server running anywhere in the world. Such a connection is appropriate for discovery but requires the author to understand which ingredient in the web-experience is which content collections in order to know what he will be able to re-use. Also, the author cannot see how the re-used content would be integrated with other content items, for example his own.

Having identified the content items of interest, he is able to identify the content project and the items that he wishes to re-use contained within. At this point, he can enter using one of the connections relations below.

6.8.2 Web-Service Content Channel

Similarly to an RSS channel, the idea of this method to connect to a content storage is to use a simple remote protocol. Using this approach, the author's learning-platform (for example an authoring-preview server as described in Chapter 3) is configured to request anything concerning the content collection through this channel. In learning platforms, there seems to be few implementations of such a method and ActiveMath's method is incomplete as it does not support both local and web-service-based connections to be run simultaneously.

The web-service approach would work fine for a small amount of queries and it would be very easy to add a new collection, since it would simply require adding the URL of the web-service to the configuration. This approach is fragile to network breaks. It is probably best qualified for a first attempt at integration of a content project within one's own project, being the first attempt at joining the net of items of the re-used collection with other nets of items.

6.8.3 Download of the Collection

Another way to load a collection which avoids all network fragility and does not need a cache is to simply download the collection in full, for example as a zip archive or through a *checkout* of a versioning server.

Following the download (and possible archive-expansion) and the configuration of the learning platform to load this content collection, it needs to be analyzed – as in, the *index* for this collection needs to be built. In ActiveMath, at least, this operation may be time consuming as explained in Chapter 4.

This connection relation, being lengthy to set up, is not appropriate for a simple discovery of the content but provides good safety. Also, it opens the door to local modifications for adaptations as we describe below.

6.8.4 Appropriation of Collections

It has been our experience that re-use is only acceptable to most authors if the right to modify is granted as well, along with the rights to redistribute the modified content. The necessity to support the modification of content collections that are re-used is important. We call this appropriation, i.e. the action of making the content collection *one's own*. To our knowledge, appropriation only works with a downloaded copy of the content-collection and is thus only possible when one has really decided to re-use a content-collection.

As has been well explained in [BSYC05], the appropriation is probably a fundamental way to have a practicing teacher make the content useful to his purposes. Re-use with appropriation could, indeed, be a good way to make potential consumers feel that something adequate is affordable for them.

In ActiveMath, locally downloaded collections can easily be appropriated provided the authoring sources are delivered alongside and the author can use the same tool.

Once appropriated, the content collection would best be re-published, in order to be offered for further re-use. This is what we turn to now.

6.9 Sharing of a Content Project on the Web

Above, we have proposed three methods to make a content and its storage available through the web. We now turn to describing the essential ingredients required by a publication on the web so that other authors can easily re-use a collection (even an appropriated collection).

A fundamental ingredient of a publication on the web is a single identifier which should not change. The first role of such a string is its use as an identifier to let the content storage identify its items, as well as to let other content items reference it. The second role is used in discussions about collections and items. It is important that people may refer to a collection or a content item anywhere in electronic communication. The third role (if care is taken that this identifier is an HTTP* URL) is to follow the *namespace document* practice of [JW04]: let the URL, when given to a browser, deliver a resource that humans using a traditional web-browser can view and let programmes make sense of it as well.

Here is the information we expect to find at such an URL:

- a license that specifies the needed rights for re-use (at best a link such as the Creative Commons Deed which provides an executive summary of the license's highlight before the legal text)
- identifiers of the content collections this one refers to: that is, the collections of all the target items of the relations in the items of this collection. These are the *other* collections that this collection needs in order to be fully functional.
- a list of all the resources involved in the collection
- a configuration of various storage connections which allows the collection to be-used (see Section 6.8)
- maybe links to a community space of the content development, to courses or other events that make use of this content collection, maybe to usage reports, potentially powered by a protocol such as TrackBack*.

All this information can be encoded within IMS Content Packaging manifests (IMS-CP*) which, by its XML nature, provides this double role of a machine processable file as well as a browser-renderable document.

6.10 Conclusions

In this chapter, we have presented several approaches to organize content collections, i.e. to publish, to store, and to serve them, in order to facilitate their re-use. This conclusion proposes a vision, illustrates the originality of our approach, and opens the perspectives for further work.

6.10.1 A Web of Versions for Peer Productions

Creating a new version of a content collection published on the web can be done in at least two ways:

- A minor version change should probably be published at the same location with simple updates of the storage. Simple web modification tracking, similar to bookmark tracking, can be used to detect the changes.
- A major version change, for example between the course of one year to another, should probably best be made as a branch, allocating a new collection URL. Such a version change is the same as building and publishing an appropriated collection with a few modifications. This can be done by anyone – this flexibility being an essential characteristic of open-source software licenses and open-content-licenses. In this way, we create a web of versions.

The web of versions has a content-package-manifest as nodes and dependencies between collections, as links. Dependencies between collections can be created by the reference of an item to an item in another collection. In ActiveMath, such links are also used in order to create content-books which are just tables-of-contents with references to items.

In this web of versions, a very important activity is the navigation between the various collections while *shopping* for the appropriate content to be re-used. For this, the ability to access a content-collection as easy as a link-click is very important. In this navigation, access may mean to simply *browse about*, that is to read the small information about it, to try it in some preview server (linked from this information), or to use it in one's own platform.

Updating a content collection, in this web of versions, can be done in several ways: if it is a minor version change, only an update of the download is needed (e.g. re-download the archive, or update using the versioning server's capabilities). A major version change requires one to change the URL: the new content collection must be connected to (e.g. be downloaded and installed), the old version probably needs to be removed, and updating references to this collection means changing the collection-identifier of each of the references.

In order to manage these identifier changes, the OMDoc language used in ActiveMath integrates the *development graph* [MAH06b] allowing each document to be endowed with `imports` elements which provide the full collection name for the short *theory-name*. Thus, references within the document can use only the short *theory-name*. Major updates of relations, involving changes to the URL of the content-collection, are done by only changing the imports, typically, one per collection. This is one of the abstraction features of the content storage described in Chapter 4.

6.10.2 Comparison to Other Approaches

We have proposed to decouple the elementary units of combination, which we suggest be as small as mathematical content items such as a a motivation, a definition, or an exercise, and the elementary units of re-use, proposing content collections which carry connection methods, ownerships, and authoring practice. This decoupling seems to be a key-point in keeping the amount of content addition operations low, thus the number of server-wide updates following such an operation. This decoupling seems, in particular, to be lacking in all learning object repositories and many approache speak about learning objects where re-use is too often considered at the individual document (or package) level [ND02], [RCM06]. An exception is the learning object repository Curriki* which has notions of composite assets.

Our approach considers the appropriation tasks at a deeper level than [MHRS06] which proposes limited re-factoring methods activated by menu entries; instead, our approach insists that arbitrary modifications should be authorable in the appropriation process and that the differencing and versioning tools can act, if needs be, to identify or visualize the modifications. Again, this task is best served, as of today, by textual source formats using source versioning tools, although some differencing tools are available for word-processing applications which are mostly able to express visual differences.

We believe that the ability for *anyone* to publish a new version, simply by putting it on the web with a description of its source, opens considerably the door to contributions. It provides full decentralization of versioning and can complement the current versioning tools which have found their place in the internal development of any content collection. Some modern versioning tools such as git* even allow versioning to be decentralized.

The Connexions* content commons seems to be closest to achieving our user-story of Section 6.1 as it provides combination and branching; it seems, however, not to support the *merging* or *return* of adaptations locally made.[5] Another major difference to ActiveMath is the granularity of the item of content re-organization: Connexions*′ modules are rather expected to be full (web-)pages, making them harder to combine.

The enhanced forms of web-links that IMS-LTI* proposes is probably the most important ingredient for re-use of content distributed in several tools. This approach, however, does not care for the distribution of authored content, for example to allow a re-using author to *take* a content project and adapt it to his need.

[5]A private communication with authors of the Connexions* project indicates a growing feature-set that allows finely detailed combination (through the course composer) and management branches and differences.

6.10.3 Open Questions

Open research questions remain, after this research, with which we conclude.

Static Web Content as Channel? It appears that, technically, the web-service channel approach could be relaxed to exploit a simple web of files using a process of compilation and map of XML content. This avenue could enable non-technical authors to publish their content-collections by a simple upload (such as FTP*) and let other authors enjoy the same *lightweight installation* that the web-service channel offers.

The right to derive in public only: our experience has shown that authors that invested body and soul in the creation of content find it hard to let anyone change their work freely and redistribute it even though the license stipulates clearly the need to quote the original authors. The most common answer to a request to add derivation right to their license is that authors wish to *know what will happen to their content*.

A potential avenue to solve this is to request the notification of usage. Global web-availability of the displayed content with links to the collections' identifiers may be sufficient to answer this: reverse links queries that most contemporary web search engines pick-up may reveal the usages; this would enable an author to see where a content project is used, to browse it, and, if need be, to request the removal or higher visibility of his authorship. A mix of legal and technical investigation is required so as to make more widespread the derivation right which is fundamental for re-use.

Measure the impact of a change: A difficult question is to empower authors that add content, or plan to do so, in order to measure the impact of their changes. Elementary methods such as easy preview under any perspective they are using (such as the user-types they use in their ActiveMath, the language, ...) exist and could certainly be enriched. For example, detecting the variations of the parameters of the perspectives and making these more easily accessible next time. However, the deeper impact is that of the evolution of the relations between the items – this needs to be measured and communicated to the user. For example, this could be changes in the course generation or learner-model propagation. Paradigms to present and query such impacts need to be researched.

The **size of a collection** is not yet much characterized: large collections are long to manipulate but lower the amount of re-use transfer operations. More importantly, collections typically attract a single online community. Should the recommendation to split collections be used?

The Right Brick of E-learning Peer-Production? Yochai Benkler in [Ben06] analyzes the ingredients that have allowed the peer productions spaces to exist and grow as large as we know them (e.g. open-source software or the WikiPedia initiatives). Among the major ingredients are the possibility of a very small contribution and its low price. We claim that the contribution of a delta posted on a static web-storage satisfy these requirements and thus believe that this framework allows decentralized peer-production.

Part III

Ease of Use

Chapter 7

Helping to Input Formulæ through Transfers

7.1 Introduction

Mathematical formulae are a fundamental part of the language of mathematics, and therefore need to be addressed with particular care by both the learners using ActiveMath and the authors creating ActiveMath content.

In this chapter we address the facilitation of mathematical input by attempting to leverage a classical paradigm: *copy-and-paste*. This is a widespread approach that allows users to select material in one place, and introduce a copy of it to another place.

By *transfer* we mean the facility offered to a user to select objects being manipulated and insert them in another place. The widespread mechanisms are:

- **copy-and-paste**: this gesture involves a view where objects can be selected, often by dragging a pointing device and showing the selection by changing the background; the selection is followed by the *copy* action which one understand often to be a transfer into a single temporary storage place called the *clipboard*; later, even after a system restart, when in an editing session which allows, input the content of the clipboard is *pasted* at the insertion point. As we see, copy-and-paste requires a selection mechanism, a clipboard, and an insertion point (all these can default to implicit values: e.g. the insertion point is the centre of a view in a graphic design programme).It should be noted that the transfers may involve translation both at copy time, where multiple flavours (the data-types of the clipboard are called flavours) can be inserted into the clipboard, and at paste time, where the *appropriate* flavour is chosen.

- **drag-and-drop**: this gesture involves a view where objects can be clicked on and physically moved in one gesture, via the use of a pointing device. The drag gesture is started when the pointing device moves while still being pressed, until a different place where the mouse-release *drops* the selected elements.Drag-and-drop is short lived and generally involves the source and target to be simultaneously visible. Content-flavour negotiation and translation happens on the basis of the flavours and the drag-action: the flavours are advertised by the *drag-source*, the drop-target chooses which one is appropriate, then the actual translations are performed.

We do not present the history of these transfer facilities, which goes as far back as the first windows and mouse-based environments.

Transferring is particularly useful for users because computers have the dual role of presenting content to be viewed and providing places to input. Among the viewed content, a considerable amount is made available by exchanges from providers or within communities; to this end the World Wide Web has been instrumental to this availability. Transfer actions are different from file copy actions because they operate on fragments of smaller size: although copy-and-paste is also applied to files in the Windows OS file explorer, it is more often used to support the copy of a sentence to insert a citation in one's writing.

7.1.1 Outline

This chapter first illustrates typical situations for learners using ActiveMath and for authors using jEditOQMath. Then we try to develop a model of our *integrated mathematical user interfaces*. After this, we describe the technological landscape where our research has taken place: the implementations of web-browsers and standards that relate to copy-and-paste or drag-and-drop.

A first facility is for the learners using ActiveMath: we investigate how to let users of a web-browser using ActiveMath transfer the terms of the mathematical formulæ presented by the system so that they can input them in places such as the exercises or search input. This was originally presented as [LJ06].

Following the implementation, an evaluation was run questioning users about how they understood the system. The results are studied and the conclusions drawn, the most important of which being that a *standard gesture* is what normal users expect and have not found for this action-type. This presents results first reported on [SPH$^+$07].

A second facility is provided to the authors: to transfer mathematical formulæ encoded in commonly-used web-places into jEditOQMath: the strategy that we implemented has been called *smart paste*, which leverages various simultaneous transformations. This was originally presented as [LAG09].

7.2 Objectives

Our investigation is at the heart of the semantic web intentions. Copy-and-paste is by far the most widely used mechanism to reproduce the construction of content and its meaning from one tool to carry it into another. Here is a quote that illustrates this desire:

[...] Having found the relevant information, you would have to copy this into yourclipboard (or the back of an envelope) and proceed to then paste in (or retype) the information into the relevant document. The type of information obtained was more often a factor of least resistance than the most relevant, timely or accurate.

T. Berners-Lee, J. Hendler, E. Miller
Integrating Applications on the Semantic Web [HBLM02]

7.2.1 Typical Transfer Situations for Learners

Mathematical documents of ActiveMath are presented to the user as part of a learning experience, and as a result of her active participation:[1] for example, she is expected to input formulæ within interactive exercises or to manage content she has learned by organizing or searching for it. In order to help the learner in her input it appears natural to try to involve *transfer paradigms* such as copy-and-paste or drag-and-drop. In the presentations she sees, content is often close to what she should input without being fully equal to it. We expect the transfer paradigm to be useful for the following typical situations:

- **Explore the object in focus:** In the process of reading examples of mathematical constructs, such as a real function or a group presentation, the learner will often want to manipulate this object further in order to explore its properties. The transfer of a representation of this object into tools such as an explorative computer-algebra-system or the function plotter can then be used (see Chapter 2 for the list of tools). Because these tools may be only capable of handling the particular formula representations and not complete expressions, the sub-term of the representation should be transferrable: for example, in the case of a reasoning in physics, only a mathematical term should to be transferred to the computer algebra system and not the units around them. In other cases, the learner will desire to transfer only a part of the formula: when she advances in he resolution of an equation, she often inserts a part of the previous step into the next.

- **Transfer to start the input:** In the process of an interactive exercise where the learner has to compute, say, the derivative of a real function, the transfer of the function from the question presentation to the places where to input appears naturally, followed by a manipulation to achieve the differentiation.

[1]We remind the reader of the conventions outlined in the introduction to name learners with the pronoun *she* while authors are named with the pronoun *he*

- **Transfer to search similar concepts:** In the process of reading or solving an interactive exercise, the memory of the learner would be helped considerably if she could find examples or exercises that she has encountered before. She would search by one of their ingredients. For example, when encountering the formula applying a singular decomposition of a matrix, our learner could transfer that formula into the search engine then replace the concrete terms by wildcards so as to search for any expression of it: this would enable her to be reminded that the expression is that of the singular decomposition even if this fact is not indicated in the first part.

Most of the content presented to the learner in ActiveMath should thus be transferrable including text fragments and terms of formulæ. The transfer should be received by places that receive formulæ: at the prompt to input formulæ within interactive exercises, of the computer algebra system, or the function plotter, or in the search function. Formulæ should be transferrable in part or in whole so that terms in large formulæ can be transferred even though the complete formulæ could not. The transfer should support consistently the extensibility offered by OpenMath*-content dictionaries.

7.2.2 Typical Transfer Situations for Authors

The situation for authors is different since their challenge is not the mathematics but the various possibilities of input.

The simple scenario that we wish to achieve is the conversion of a mathematical formula on a page found on the Web to a source document for the ActiveMath learning environment. For example, Wikipedia (`http://www.wikipedia.org/`) or MathWorld (`http://www.mathworld.com/`) are both web-sites with a rich wealth of mathematical content. In many situations it is possible to convert the content from these pages for re-use, e.g., in educational activities. We wish to empower the authors to do this.

We propose to leverage multiple converters that are already available within one simple copy-and-paste action which we shall call *smart-paste*.

The research on smart-paste follows a common dialogue we, as trainers of the usage of jEditOQMath, had with future ActiveMath authors. We were presenting them the features of jEditOQMath and were often asked "So you are supporting TeX?": So far, this question could only be answered negatively since *supporting TeX** meant encoding many formulæ which are not semantically interpretable; it also required jEditOQMath to support all sorts of macro-contexts for which we would have needed to implement almost all of TeX. The smart-paste approach presents a modest way of supporting TeX.

7.2.3 Integrated Mathematical User Interfaces

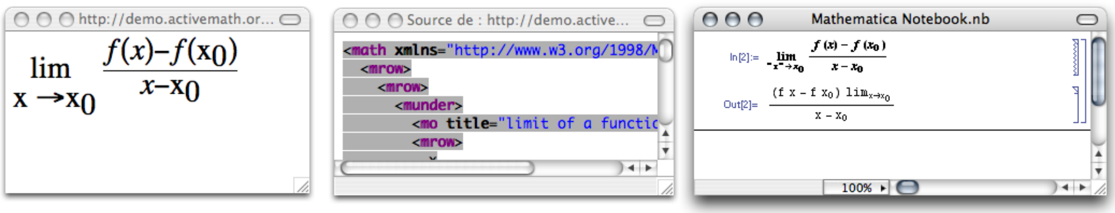
When designing the ActiveMath learning environment a series of design design concepts were discussed in order to ensure a truly *integrated* user-interface for the manipulation of mathematical content. These principles are outlined below:

- The platform should be able to present mathematical content graphically with a quality approaching classical print. If the platform is intended for use in multiple languages, this presentation should be adapted to the specific customs of mathematical notations within a particular region.
- The mathematical notations should look the same irrespective of where within the platform they appear or whether they are presented on screen or via a printout.
- The mathematical notations should, ideally, be enhanced by interactive features which should support readers' memory about the meaning of graphical constructs. Tooltips or hyperlinks can provide this interactivity.
- All formulæ in the platform should use notations that look similar. Expressions input by the user should be rendered using the same appearance so that the user understands a common-language between presentation and input.
- As much as possible, presented formulæ should be transferrable to input areas following a paradigm familiar to the user.
- The mathematical expressions should be processable by all tools offered by the platform. For example, if a function plotter is offered, it should be possible to plot the graph of most functions found within the presented content. Similarly, a search tool should match the presented content in a consistent way with the input queries.

Several of these requirements are also expressed in [DH03].

The range of actions that can be performed on mathematical expressions defines the depth to which the semantics of the expressions needs to be represented and processed by the platform. It can range from the simple rendering function to such deep processing as function plotting, evaluating computations in a computer-algebra-system, or type checking for the expressions processed by the platform.

Orthogonally, platforms can be differentiated with regards to their semantic breadth, that is, the mathematical domain that could be covered by the tools. These can range from tools dedicated to a particular task involving particular manipulations (e.g. operations on matrices as in [SS06] or homological spaces as in [HPRR10]) to generic repositories which can offer services such as search on any mathematical formula but not other processing as described in the related systems in Chapter 5.


http://demo.activemath.or...
$\lim_{x \to x_0} \frac{f(x)-f(x_0)}{x-x_0}$
Source de : http://demo.active...
Mathematica Notebook.nb
In[2]:=
Out[2]=
100%

```
<math xmlns="http://www.w3.org/1998/M
  <mrow>
    <mrow>
      <munder>
        <mo title="limit of a functic
        <mrow>
```

Figure 7.1: A surprising translation from presentation to content: A *difference quotient expression* in MathML-presentation is on the left. For it to be copied one needs to copy its source which can then be pasted into Mathematica which recognises automatically. However, the semantic interpretation of Mathematica, obtained by pressing the return key, is somewhat surprising! The same procedure in Maple yields an error at interpreting the limit.

7.2.4 Other Platforms with Integrated Mathematical User-Interfaces

Quite a few platforms implement, in some way, the principles of integrated mathematical user-interfaces we have described above.

On the desktop, mathematical graphical composition engines can be considered to implement these principles. Examples of composition engines include Scientific Workplace,[2] XThink's MathJournal,[3], or TeXmacs.[4].

In these tools, the mathematical expressions are mostly manipulated in presentation format, only when it comes to computing with them, an engine converts the attempted bits to the necessary form and warns the user if this fails. This conversion is rarely extensible; moreover, it does not support multiple traditions of mathematical notations. These platforms support keyboard input and palette-based editing, with XThink even supporting stylus input.

These tools all have particular variants of the paste command which the user has to choose a priori. We claim that this is error prone and could be better formulated in such a way as the smart paste approach we describe below.

The classical computer algebra systems such as Maple, Mathematica, or MuPad,[5] all have a similar approach with a stronger orientation of computable expressions; they all support multiple encodings of a formula (presentation-2d, TeX*, tool-specific source, ...) with automatic translations between them, but for a single (North American) culture. The naïvety of the translation brings fancy surprises such as the translation from MathML*-presentation expressions in the Figure 7.1.

To our knowledge, none of these tools allow a user to search through libraries of content especially for mathematical expressions. They can be used for e-learning

[2]For the Scientific Workplace family of products, see `http://www.mackichan.com/`.
[3]See `http://www.xthink.com/Products.html` for MathJournal.
[4]See `http://www.texmacs.org/` for TeXmacs.
[5]See `http://www.maplesoft.com/`, `http://www.mathematica.com`, and `http://www.mupad.de/`.

activities with interactive exercises but the authoring of such content often requires platform-specific programming.

This feature is probably closest to the *smart paste* approach which we describe below. However, the ability to involve a decision of the user in order to guide the translation process appears to not have been explored, as of yet.

On the Web, few tools try to apply the integrated approach. We are only aware of the Wiris computer-algebra-system[6] which is a complete applet*-based computer-algebra-system, with computations relayed to a server but its copy-and-paste functionality is internal only.

As far as we know, all these tools permit input of mathematical formulæ within the text of a broader document, similarly to word-processors. However, the buttons and menus that allow this input are located far off within the boundary of the window of the document. This seems to contrast with the implementation described below whose evaluation has shown challenges to the usage of an independent formula editor window, which is taken out of context such as the one we have used in ActiveMath.

7.3 A History of Browser Capabilities for Transferring

The research investigations described in this chapter have been performed in three eras which are characterized by different transfer mechanisms available in the web-browsers' technology available at that time.

We have focussed our work on 3 major web-browsers:

- *Mozilla Firefox*: this browser emerged from the Netscape suite, which, in turn, emerged from the browsers of the web pre-history (NCSA Mosaic). Our investigations have have covered versions of FireFox from FireFox 1.0 in 2004 to FireFox 3.6, in 2009. See `http://www.mozilla.org/`. Its support for MathML*, sometimes with the need for extra fonts, and its cross-platform-ness has made it the browser of choice of the ActiveMath learning environment.
- *MicroSoft Internet Explorer*, from versions 6 in 2002 till 9 in 2010 is recognized as a major browser in the world statistics. See `http://www.microsoft.com/explorer/`. This browser has been a low priority for the ActiveMath group with recurring issues that have taken time to be solved; moreover, an add-on is needed to render a web-page with MathML*, called MathPlayer.

[6]See `http://www.wiris.com` about the Wiris CAS.

- *Apple Safari*, from version 1.0 in 2003 till version 4.0 in 2009, has only been in focus of the ActiveMath group from 2008 onwards. Based on the same core, a recent cousin of Safari appeared in 2009 called Google Chrome; as of this writing, it is leading the user-shares of Mozilla. See http://www.webkit.org/.

Several other browsers have been experimented with during this research, notably Opera, but have not been the main focus of our research.

For all the browsers, one could delineate the following eras by looking at the abilities of users to perform transfers using browsers:

The **early era** is marked by an absence of possibility, for web-page creators and web-server makers, to have any control of what can be copied by the user of a web-browser beyond the presented text or image. In this era, one could copy-and-paste plain and styled text and images, one could drag-and-drop these as well as hyperlinks. Pictures also could be copied to text using their *alt* attribute's value.

In this era MathML* expressions could, once right-clicked, be *copied to clipboard*, but only in their entirety and as source in plain text. This was in the spirit of the computer algebra systems of the time.

This era ranges, for FireFox, from version 1.0 till version 3.5, for Internet Explorer from version 6 till version 9 where the untrusted web-paged can almost nothing; it does not concern Safari.

The **API era** is characterized by the birth of application programming interfaces (API* s) for each of the web browsers that allow the scripts to somewhatinfluence what is being copy-and-pasted.

Because the URLs to API* specifications have had a strong tendency to change (MicroSoft and Apple being the most frequent URL changers) we have tried to maintain a web-page about the literature around copy-and-paste: `http://eds.activemath.org/en/transfers-literature`.

In this era, Internet Explorer, from version 5, offered an API* to allow web-pages to read from and write to the clipboard; unfortunately, it has been quickly limited to only fully trusted sites (such as intranets) because it would otherwise have allowed any web-site to read from the clipboard without the user knowing. Multiple newspapers warned users about this threat.

Apple Safari came, from day one, with an API* that could only modify what is being copied to the clipboard at copy time and what was pasted at paste time; that is, when the user actually invoked the copy or paste commands. Though not very visible, this API* has been secure all the time because it was based on the standard gestures and was not invokable by an untrusted script (for example by a button labeled "copy train schedule").

In these well implemented API* s, the flavours[7] that were allowed to be written to the clipboard were quite limited - allowing only plain-text, styled-text, images and URL-lists.

The **API standardization era**: as they became aware of the interest for web-site makers to offer rich copied sources, the webAPI working group of the W3C, under the lead of Charles McCathie-Neville, issued a first working draft of *Clipboard operations*, `http://www.w3.org/TR/2006/WD-clipboard-apis-20061115/`. This seemed to reach some consensus but has not received multiple implementations.

The group behind HTML5 included a section on drag-and-drop and copy-and-paste inspired by the specification of Internet Explorer which enabled, theoretically, the inclusion of arbitrary data-flavors. As part of a global trend of implementing HTML5, FireFox 3.6, Safari 4.0, and Internet Explorer 9 implemented this section. See the Section 7.7 on drag-and-drop at `http://www.w3.org/TR/html5/dnd.html#dnd`.

The webapp working group of the W3C restarted work on the clipboard operations in 2010, under the lead of Hallvord Steen. Several drafts appeared in 2010 and 2011. See `http://www.w3.org/TR/clipboard-apis/` – we hope to see implementations that allow extra content-types. We refer to the future-work section, 7.7.1, for an outline of the possibilities provided by these API* s.

In parallel to that, based on a written agreement between the Design Science and MicroSoft corporations, MathML* 3's Chapter 6 (which I edited) included in 2010 a section on how to put MathML* to the clipboard, and how MathML can be written to describe alternative representations of the same object which should go into the clipboard using `semantics` and `annotation` elements; see [CIM10, Chapter 6].

Unfortunately, only the first part has been implemented by web-browsers and contemporary systems because offering the ability to put arbitrary flavors within the users' clipboard may let the user paste a somewhat untrusted piece of content within a trusted environment which, hence, would gain greater privileges (e.g. with a call to local or remote servers by embedding an inclusion of an image).

We refer to `http://w3.org/Math/testsuite` to see reports of testing this feature (section *Clipboard* within the section *General*). To date, only the Wiris Input Editor of ActiveMath implements the copy of alternate representations provided by annotation elements.

[7]It has become classic to call flavour the type of data involved in copy-and-paste or drag-and-drop operations. Flavours are often comparable to media-types.

7.4 ActiveMath's Browser Transfer for Learner

We now present the implementation in ActiveMath of the transfer of mathematical formulæ aimed at students. It uses a contextual menu and the drag-and-drop paradigm. The implementation described here has been realized in 2005 and 2006 with the main target being Mozilla 1.7 and FireFox 1.5.

Formulæ delivered to the browser of a user of ActiveMath are generated from OpenMath* using the presentation architecture, an authorable rendering engine for documents with mathematical formulæ described in Chapter 2.

The rendering output, for formulæ, is done either in HTML* (with CSS*), MathML*-presentation, or TeX* (finally rendered to PDF*). The semantic of the formulæ encoded in OpenMath* is presented with added-value features:

- tooltips indicating the (localized) name of each symbol when the mouse is on the operators of this symbol,
- links to the search for the symbol's definition upon clicking on the presented symbol,
- in the case of HTML* +CSS*, a highlight of the presentation of the smallest subterm behind the mouse. Figure 2.7 shows such a presentation.

This highlighting is performed by decorating, in the pre-processor phase of the presentation pipeline, each of the source's `OMOBJ` elements (the root of each formula) with the identifier of the containing item followed by its order-number and the XPath coordinate of each OpenMath* application elements (`OMA`). These coordinates are passed along the transformation as `id` attributes of elements enclosing the rendering of each term. When presented in the browser, a script can use the elements hierarchy as well as the `id` attribute-values to change the background colour of the rendering of the smallest sub-term term under the mouse which is assumed to be the smallest OpenMath* sub-tree containing the symbols' rendering. An illustration of these annotations and their actions is in Figure 7.4.

When the user clicks on any part of the formula, the smallest term containing the rendering under it is highlighted and a contextual menu appears as depicted in Figure 7.3. It offers to:

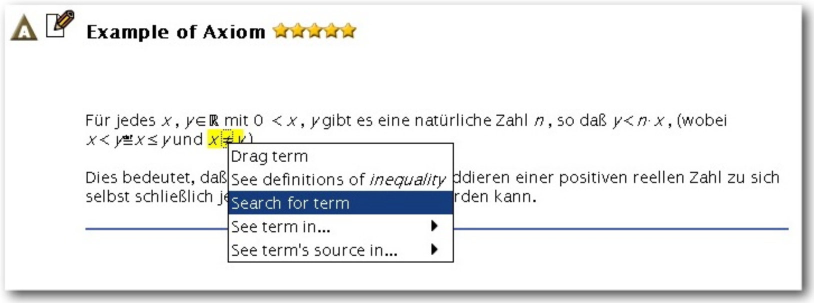
- search for the indicated term in the ActiveMath search tool (see Chapter 5),
- search for definitions of the presented symbol that was clicked,
- display this term in MathML* or OpenMath* either in original form (which potentially contains many unknown symbols) or translated to have terms in appropriate OpenMath CD-groups if possible. This display is delivered by the *clipping-controller*,

```
<OMOBJ version="2.0" struct-node-kind="ground"
      struct-node-path="unmb-mbase://LeAM_calculus/deriv/ex_diff_parab@/CMP[17]/OMOBJ[0]">
      <OMA struct-node-path="/OMA[0]">
      <OMS cd="relation1" name="eq" struct-node-path="/OMS[0]" />
    <OMA struct-node-path="/OMA[0]">
      <OMS cd="arith1" name="minus" struct-node-path="/OMS[0]" />
      <OMA struct-node-path="/OMA[0]">
        <OMS cd="arith1" name="power" struct-node-path="/OMS[0]" />
        <OMA struct-node-path="/OMA[0]">
          <OMS cd="arith1" name="minus" struct-node-path="/OMS[0]" />
          <OMV name="x" struct-node-path="/OMV[0]" />
          <OMA struct-node-path="/OMA[0]">
            <OMS cd="basics_symbols" name="x_coord" struct-node-path="/OMS[0]" />
            <OMV name="K" struct-node-path="/OMV[0]" />
          </OMA>
        </OMA>
        <OMI struct-node-path="/OMI[0]">2</OMI>
      </OMA>
      <OMA struct-node-path="/OMA[1]">
        <OMS cd="arith1" name="power" struct-node-path="/OMS[0]" />
        <OMA struct-node-path="/OMA[0]">
          <OMS cd="arith1" name="minus" struct-node-path="/OMS[0]" />
          <OMV name="x__0" struct-node-path="/OMV[0]" />
          <OMA struct-node-path="/OMA[0]">
            <OMS cd="basics_symbols" name="x_coord" struct-node-path="/OMS[0]" />
            <OMV name="K" struct-node-path="/OMV[0]" />
          </OMA>
        </OMA>
        <OMI struct-node-path="/OMI[0]">2</OMI>
      </OMA>
    </OMA>
    <OMA struct-node-path="/OMA[1]"> [58 lines]
  </OMA>
</OMOBJ>
```

Figure 7.2: OpenMath source of the term that is dragged in Figure 7.4: the term surrounded by the second `OMA` element of this example (the other part of the equality is *folded*.). The coordinates are in the `struct-node` attributes indicating the type and coordinates of each sub-term. The pre-processor inserts these as well as `xref` attributes on each `OMS` element which allow these elements to function as absolute references as described in Chapter 4. The coordinate attributes are exploited by the XSLT transformation, concatenating the values of the `struct-node-path` attributes for each of its ancestors: these yield a coordinate the *clip-controller* can render in multiple flavours as requested.

- prepare the term-highlight for later drag, which draws a black box around the sub-term by applying a layer on top of the term. The content of this layer is entirely draggeable because it is an anchor. This link's URL points to the *clipping-controller*, which delivers the term in various formats.

If this link is dropped onto the Wiris Input Editor (the default formula editor in ActiveMath), the input editor requests from this link and inserts a copy of the formula. This sequence of actions is depicted in Figure 7.4 and the OpenMath* tree of the term is presented in Figure 7.2.



Example of Axiom

Für jedes x, $y \in \mathbb{R}$ mit $0 < x$, y gibt es eine natürliche Zahl n, so daß $y < n \cdot x$, (wobei $x < y \Leftrightarrow x \leq y$ und $x \neq y$)

Dies bedeutet, daß ddieren einer positiven reellen Zahl zu sich selbst schließlich j rden kann.

Figure 7.3: The contextual menu on a term.

7.4.1 Receiving and Transferring Formulæ

We have described how a hyperlink is dragged out to represent a term. The receiving applications, currently the Wiris input-editor [MEC+06] integrated in ActiveMath, receives drops of URIs and launches the corresponding HTTP* GET request. This request indicates the supported media-types, the Wiris input editor indicates the OpenMath* media-type*[8]. The media-type alone is indicated as well as variants of it enriched by a parameter indicating the supported CD-group, a set of content-dictionaries indicating the supported symbols. For example, the media-type of an expression that would be translated well into MathML*-content would be with the symbols of the MathML-CD-group (declared in [BCC+04]:
`application/xml+openmath;cdgroup=http://www.openmath.org/cdgroups/mathml.cdg`

The *clip-controller*, when reached, extracts the OpenMath* content from the content store and tries to apply the translation to obtain a term with symbols only in the desired CD-group. If that fails it tries the next supported content-type or returns an error. The translation is operated by the declaration of the relevant CD-groups together with *rephrase-rules* which are pairs of OpenMath* expressions mapping one to the other and are input by authors declaring new symbols. An example of such a translation is for the symbol `unary-plus`: this symbol has been avoided as part of the standard OpenMath* content-dictionaries, and therefore would not be understood by a traditional OpenMath processor. Bu it is important for presentation purposes. The following rephrase rule thus states that it can be removed when going to the MathML CD-group:

```
<rephrase> <!-- maps +a to a -->
    <OMOBJ>
      <OMA>
        <OMS cd="basics_symbols" name="unary_plus"/>
       <OMV name="a"/>
```

[8]The OpenMath* media-type is not actually registred in the central authority at IANA but the usage `application/xml+openmath` follows naturally from the RFC-3023 [MSK01].

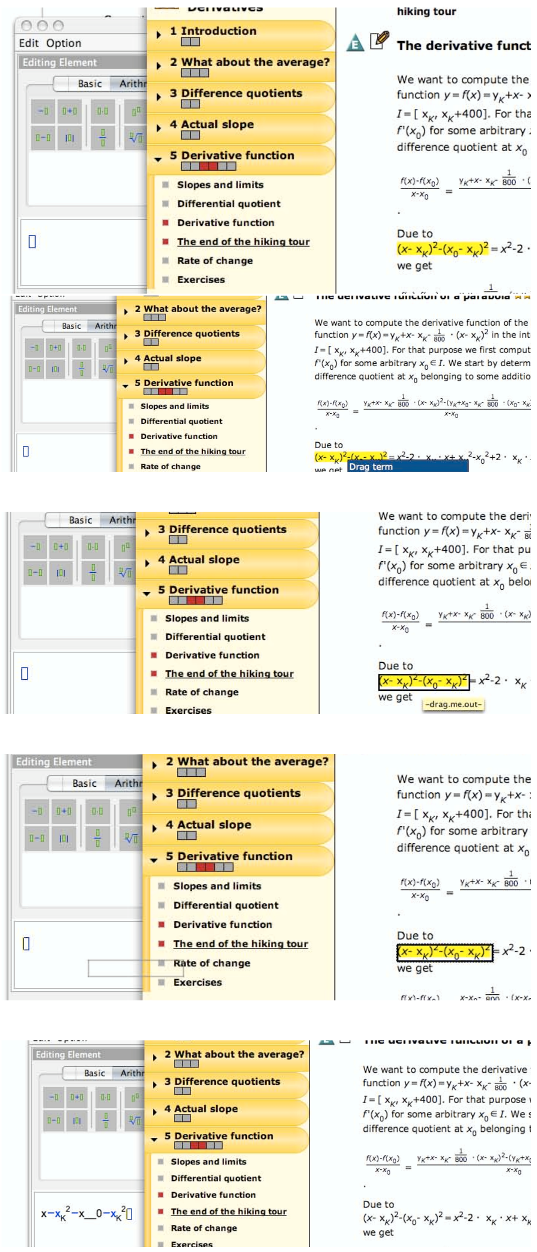


Figure 7.4: The steps of the drag-and-drop of a term: hover the mouse to see the term, right-click, invoke drag-term in the context menu, drag-it, drop it in to the input editor.

```
      </OMA>
    </OMOBJ>
    <OMOBJ>
      <OMV name="a"/>
    </OMOBJ>
  </rephrase>
```

The translation procedure can also go to presentation media-type*. In this case, it is concluded by an XSLT* transformation and a *compilation* which provides the visual formats indicated above (MathML*, TeX*, HTML*...) offered by the presentation pipeline (see Section 2.4.7).

The Wiris Input Editor is used in the exercises [GPM05] and search tools as a formula editor with palettes of choosable symbols. This editor is extensible by a set of *domains* which provide the notations associating symbols with their rendering. So as to to enable that any expression rendered by ActiveMath is also rendered in the Wiris Input Editor, the notations described in Chapter 2 for the symbols which do not already have a notation in the domains, are also exported to the Wiris domain.[9]

Drops can also go to input places which cannot be so easily extended such as the function plotter used in ActiveMath. It should be noted that the transfer to the function plotter is an illustration of two specialties:

- Sub-terms to be transferred are not just complete formulæ since functions presented in texts or exercises are very often presented as equations and not as simple functions.
- The plotter is not extensible in its mathematical core, but it only does computations (as opposed to re-representation). As a result the transfer is extensible to expressions with new symbols as long as these can be converted to computable expressions using the basic set of symbols. Although incomplete, rewrite rules can leverage the usage of particular symbols that are close to functions; it would support a teacher defining new symbols for the sake of a different naming convention.

Drag-and-drop to the function plotter is now at the prototype stage and has been implemented by an XSLT*-stylesheet on the received OpenMath* term which produces the linear input of the plotter component. When the drop is successfully converted to a function in one variable, a graph is automatically drawn. The transfer can fail in some cases, in particular when more than one variable is used; in this case the result of the transformation can still be hand-edited (e.g. replacing the occurrences of all but one variable by constants).

External desktop applications could be recipients of this transfer in two ways:

[9]This export feature has been experimental for many years and seems to cause usability issues in the input-editor; it has been often disabled.

- Applications receiving a drop of *clip* URLs could fetch its content over an HTTP* GET. The ability for HTTP requests to specify the supported content-types as well as the user-agent identifications leaves space for such a communication to be highly tuned.

- Using the functions of the formula context menu, the user can also select an appropriate format of delivery; this action would be followed by the invocation of the copy-and-paste actions. For the presentation-oriented applications such as graphical programmes or word-processors, the presentation formats such as MathML*, HTML*, TeX* or PDF* can provide good support. This is the approach suggested by the widespread MathJax module.[10]

7.5 Evaluation of the Browser Transfer of ActiveMath

The LEACTIVEMATH project, which ran from 2004 till 2006, included classroom evaluations. The system has been tested in class-based learning in about 10 classrooms in Gymnasium (ages 16-18) classes around Augsburg, in the University of Malaga, and in the University of Edinburgh during the Fall semester 2006-2007. The objective of the evaluation was to measure the impact and affordance of learning with the learning environment. A detailed evaluation has been published in [SPH$^+$07]. A quick analysis of the logging data of the usage of the main evaluation server `http://leam-calculus.activemath.org/` has indicated, in the period 2006-2007, a quantity of 31994 formulæ input in linear syntax and 22407 input with the input editor.

Optional post-session surveys have estimated a mean quality of formula presentation of 66%, in comparison to 72% quality of texts. It should be noted, that in order to provide the interactive support on formulæ as described above, the default presentation medium was still HTML* +CSS* which explains why formula rendering could be enhanced.

Among the feedback that we obtained, was the *betrayal* of the consistency rule indicating that the visual presentation of the formulæ being input should be the same as the re-rendered content: for example $\sin x$, although written without parentheses in rendered formulæ, does need the brackets in the input since the latter needs to know when the argument of sin is finished. This is an example of an irreducible distance between an input language and presentation language.

[10]MathJax is a recent JavaScript* which aims at rendering formulæ in all contemporary web-browsers. See `http://mathjax.org/`.

7.5.1 In-depth Evaluation Tasks

This learning experience was complemented by guided tasks with questionnaires to allow a precise evaluation about specialized facets of the system. The evaluation tasks focussed on various aspects of ActiveMath: access and presentation of content-items, the exercise system, the learner-model, and the tutorial component. We focus on the evaluation tasks of the mathematical input which were taken by 70 students of the University of Edinburgh and the University of Malaga in December (in various undergraduate fields, such as mathematics, engineering, or economics).

The main goal of these tasks was to evaluate the ease-of-use and preferences of input of mathematical formulæ in exercises. Subjects were set a series of derivation exercises and asked to answer the exercises using a variety of input methods. They then provided feedback via post-task questionnaires. The formulæ to be input for each exercise were provided graphically in the task descriptions; it was possible to input them using the Wiris Input Editor or using the textual input-box, a Maple-like syntax textual input.

The following tasks were given:

1. The first three tasks encompass an easy exercise and require the subject to input the same polynomial $\frac{70}{3} \cdot x^4 + \frac{16}{3} \cdot x$ using a variety of methods. In the first task the subject has to input the polynomial using the plain-text syntax.
2. The second task requires the polynomial to be input using buttons and keys in the input-editor.
3. The third task requires the subject to drag-and-drop the polynomial from the original question into the input-editor (as is pictured in Figure 7.5).
4. The fourth task occurs within an exercise of medium difficulty and lets the student choose which of the three input methods (text, input-editor, or drag-and-drop) they wish to use to input $2 \cdot x \cdot (6 \cdot a \cdot x^2 + b)$.
5. Similarly the fifth task lets the subject choose how to input $15 \cdot \cos(x)^4 \cdot (-\sin x)$ into an exercise of medium difficulty.
6. Finally, the sixth left the students free to input $\cos(7 \cdot x^{11} - 3 \cdot x^3) \cdot (77 \cdot x^{10} - 9 \cdot x^2)$ within a difficult exercise.

Some of the exercises have been dynamically generated. As a result, students had slightly differing evaluation tasks.

The choice of using the input-editor can be decided at any time during the first input of an exercise.

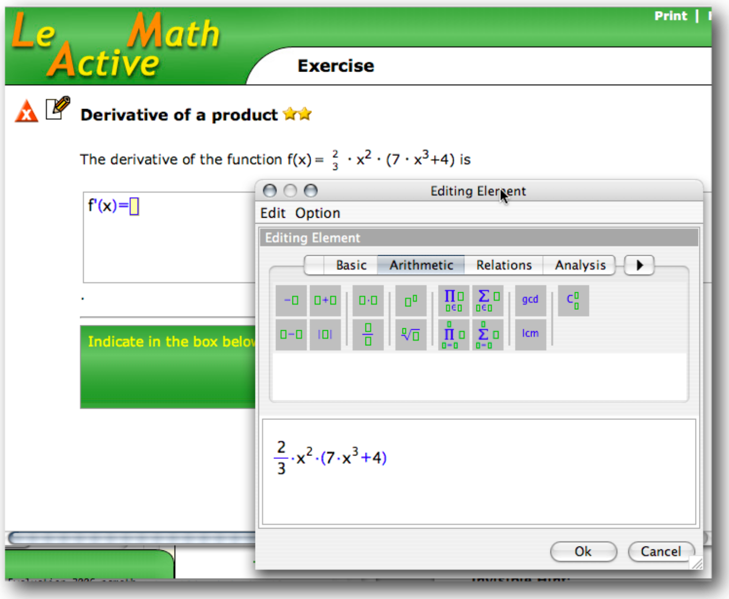


Figure 7.5: The third input-task: using the method of first drag-and-dropping the term to be derived then modifying it.

7.5.2 Results and Discussion

A few conclusions follow directly from the results of this evaluation:

- Most users find the answers simple enough to type into the text-field (68%) and typing into the field is the best solution for all of the free-input tasks: text input was rated as the best input method by 68% of all subjects for task 4, 55% for task 5, and 60% for task 6. The increasing difficulty and length of formulæ across the three tasks which was expected to encourage the usage of the input-editor and drag-and-drop but had no such effect.
- Most users believe that the input editor is too complex (63%). However, conflicting with their actual behaviour, they believe that the input-editor or drag-and-drop would be more suited for more complex expressions (63% and 71% respectively).
- Most users knew where to find the buttons needed to input separate symbols (70%) among the many tabs of the input-editor palettes. However, most users bypassed the buttons and instead typed directly into the input editor, relying on the editor to convert their text input into the appropriate notation.
- Most students found the input-editor's syntax-checking, based on the simple-type-system [Dav00], to be a useful feature. However, subjects reported

that they were often unable to successfully input expressions as the input-editor would reject the expression due to text-input syntax errors without identifying what the error was in a way that would enable them to solve it (repeatedly stating *Input syntax error* does not help a user who cannot self-diagnose their error).

- One of the major issues of the input-editor reported by subjects as extra comments at the end of the evaluation was the problem of sharing the screen space between the main ActiveMath webpage and the pop-up Input-Editor Java* applet*. In order to successfully keep an eye on or drag content from the webpage to the Input-editor both windows have to be simultaneously visible. This is often not possible due to restricted monitor size, resolution, and the large size of the input-editor. This impracticality, combined with inexplicable syntax errors, loading time and storage-related issues meant that the subjects' experience of using the Input Editor was less than enjoyable.
- The fact that errors are properly flagged and resolvable appears to be very important in a learning situation. As an example, we quote one of the experimentation subjects: *It didn't accept my answer as correct, despite clear syntax (checked by the syntax checker) and my answer being correct*. This sentence can certainly show the state of mind of the learner almost losing confidence in his conceiving, writing, and inputting capabilities, which are all bound together at this moment.

The usage of the drag-and-drop gesture has been the sole possible *transfer* gesture that we could offer within the (untrusted) web-based environment of ActiveMath. This gesture is more difficult than a *standard* copy-and-paste that can be done with normal selection highlighting and the platform clipboard.

The task-evaluations have provided the following results:

- 60% users find that drag-and-drop was difficult to use, 60% found highlighting frustrating, and 70% would have not known they could drag a formula.
- 70% of the users find drag-and-drop *a clever way of avoiding the syntax problems* and more than 60% would use it to drag-and-drop from exercise questions, book-pages to pages of exercises, search tools, or computer-algebra system. But only 55% would use it to drag to outside applications (such as word-processors).

These results tend to confirm a general wish for transfer facilities but a weak acceptance of the non-standard selection and transfer mechanisms that we provided in LEACTIVEMATH. Moreover, the fact that drag-and-drop requires the source and target to be almost simultaneously visible is probably an impediment to easy transfers.

We dare to conclude from this evaluation that the standard gestures need to be used; this was not possible at the time. The API*, and especially the API*-standardization era (as we have described in Section 7.3) are the eras that were needed to achieve this. See the future works Section 7.7.

7.6 Smart Paste to Help Authors Input Formulæ

7.6.1 Motivation

As we have described in Chapter 2 and in Chapter 6, ActiveMath content consists of sets of OMDoc files called *content collections* which share a practice of authoring. Most of the collections are based on the OQMath format described in Chapter 3: this format is mostly made of a readable OMDoc encoding with mathematical formulæ written in compact linear form and compiled to OpenMath* by the QMath processor [Pal06]. This processor uses *notation contexts* which define new notations using Unicode characters. Very often an extra notation context is attached to a collection which includes the standard notation of other contexts as is done in the content collections created by the templates of jEditOQMath.

In this research we investigate how it is possible to let the authors exploit the converters that are widely available between MathML*-presentation, TeX*, MathML*-content and OpenMath* to transfer mathematical formulæ from a remote source to the QMath syntax of the current collection.

7.6.2 Definition

We define *smart paste* as the action of transferring what the user has put in the clipboard (what he understands to have put) to an editing zone using a transformation appropriately chosen for the editing task at hand. Smart paste may involve interactions with the user in order to guide the program in choosing the best alternative. That guidance should, as much as possible, be leveraged to minimize interaction requests in subsequent smart paste invocations since it is expected to be used repetitively.

7.6.3 Smart Paste of Formulæ in jEditOQMath

jEditOQMath has been extended with a smart paste functionality, it has been assigned to an alternate shortcut, similar to word processors' "special paste": CTRL-SHIFT-V or, on MacOSX, CMD-SHIFT-V.

The smart paste of jEditOQMath tries to convert formulæ from:

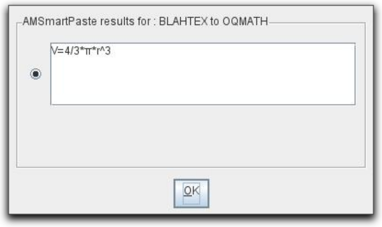
- the TeXvc syntax of Wikipedia (which is very TeX like)
- MathML*-presentation
- MathML*-content
- OpenMath*

to the QMath format, using the notation context of the file it is pasted into.

A few *content sniffers* are used in order to detect the format of the input which, thus far, is still in the plain text media-type*. The determined formats are then used to choose the conversion pipelines which are made of the following ingredients:

- MathML*-presentation to MathML*-content, in a fixed and comprehensive way, as provided by the WebEQ developer tools[11]
- Content MathML* to OpenMath* as provided by the stylesheet of David Carlisle [12]
- MathML-presentation to OpenMath as provided by the Wiris OpenMath tools [MEC+06], in a way that can be configured by the *notation domain*
- TeXvc to MathML-presentation through the usage of the blahtex command-line tool[13]
- OpenMath to QMath thanks to an XSLT* stylesheet that is generated on the fly from the notation context of the current file, outputting the necessary brackets (as little as possible).

7.6.4 Example Smart Paste Usage


AMSmartPaste results for : BLAHTEX to OQMATH

V=4/3*π*r^3

Figure 7.6: Pasting the formula for the volume of a ball from Wikipedia

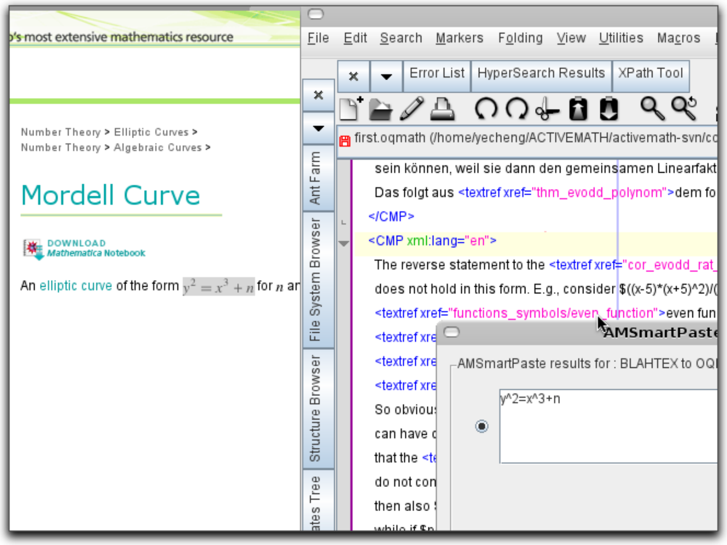
One of the first targets is to use the Wikipedia web-site whose mathematical content is ever growing. Most of the formulæ are encoded using the TeXvc syntax and converted to pictures. Copying the formula picture using Firefox puts the content of the alt-attribute of the image into the clipboard, that the TeXvc processor kindly fills that attribute with the source. For example, many formulæ on `http://de.wikipedia.org/wiki/Formelsammlung_Geometrie` can be easily copied using the converter chain BlahTeX - WebEQ - CmmlToOm - OpenMath 2 OQMath

- $A = \frac{g \cdot h}{2}$
- $V = \frac{4}{3}\pi r^3$
- $h^2 = q \cdot p$

[11]See `http://www.dessci.com/`.

[12]See `http://www.openmath.org/cdfiles2/xsl/om2cmml.xsl` for the stylesheet Pragmatic-Content-MathML* to OpenMath*

[13]See `http://gva.noekeon.org/blahtexml/` for the BlahTeX converter from TeXvc-syntax to MathML presentation.


's most extensive mathematics resource
Number Theory > Elliptic Curves >
Number Theory > Algebraic Curves >
Mordell Curve
DOWNLOAD Mathematica Notebook
File Edit Search Markers Folding View Utilities Macros
Error List
HyperSearch Results
XPath Tool
Ant Farm
File System Browser
Structure Browser
ates Tree
AMSmartPaste

An elliptic curve of the form $y^2 = x^3 + n$ for n ar

```
first.oqmath (/home/yecheng/ACTIVEMATH/activemath-svn/co
  sein können, weil sie dann den gemeinsamen Linearfakt
  Das folgt aus <textref xref="thm_evodd_polynom">dem fo
</CMP>
<CMP xml:lang="en">
  The reverse statement to the <textref xref="cor_evodd_rat_
  does not hold in this form. E.g., consider $((x-5)*(x+5)^2)/(
  <textref xref="functions_symbols/even_function">even fun
  <textref xre
  <textref xre
  <textref xre
  So obviou
  can have c
  that the <te
  do not con
  then also $
  while if $n
```

AMSmartPaste results for : BLAHTEX to OQ

● y^2=x^3+n

Figure 7.7: Transferring the equation of the Mordell curve from MathWorld

- $O = r^2 \cdot \pi + \pi \cdot r \cdot s = r \cdot \pi \cdot (r + s)$

A similar gesture is done from MathWorld – for example the generic formula of Mordell Curves at `http://mathworld.wolfram.com/MordellCurve.html` presented in Figure 7.7: the alt-tag of the pictures is used again.

The case of JS-math powered websites is supported by this software's inbuilt ability to show the TeX-like source of each formula when one double clicks on it. The page `http://planetmath.org/encyclopedia/GoniometricFormulae.html` offers a wealth of formulæ to test where a few succeed.

7.6.5 Failures of Smart Paste and their Recovery

Not all formulæ succeed on the page mentioned in Figure 7.8 and our heuristics certainly need to be boosted to raise the success rate.

A common style of errors is the inclusion of the punctuation at the end of a formulæ as in the figure on the right: the markup for such a picture includes the period inside the formula element (be it in TeX* or MathML*). Not surprisingly, this breaks all conversions from presentation to content since it corrupts the mathematical semantics of the formula. A safe way to remove such punctuation is desirable and will probably be best placed within a process of MathML*-processing.

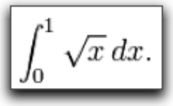
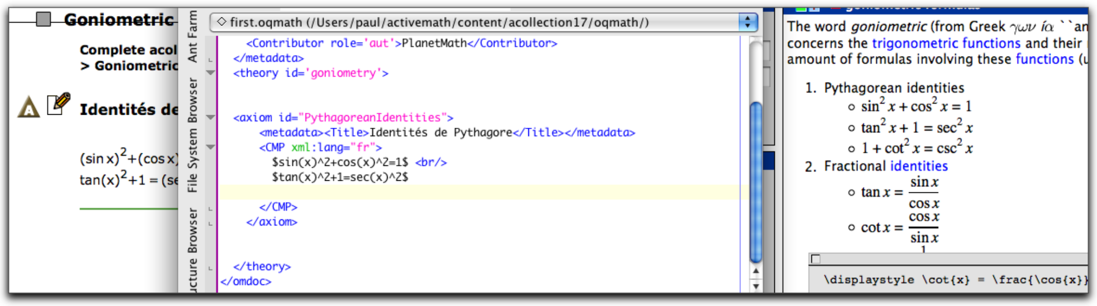


Figure 7.9: formula ending with a punctuation.



$(\sin x)^2+(\cos x)$
$\tan(x)^2+1=(se$

first.oqmath (/Users/paul/activemath/content/acollection17/oqmath/)

```
    <Contributor role='aut'>PlanetMath</Contributor>
  </metadata>
  <theory id='goniometry'>


  <axiom id="PythagoreanIdentities">
      <metadata><Title>Identités de Pythagore</Title></metadata>
      <CMP xml:lang="fr">
        $sin(x)^2+cos(x)^2=1$ <br/>
        $tan(x)^2+1=sec(x)^2$

      </CMP>
    </axiom>


  </theory>
</omdoc>
```

The word *goniometric* (from Greek $\gamma\omega\nu$ $i\alpha$ ``an
concerns the trigonometric functions and their
amount of formulas involving these functions (u

1. Pythagorean identities
   - $\sin^2 x + \cos^2 x = 1$
   - $\tan^2 x + 1 = \sec^2 x$
   - $1 + \cot^2 x = \csc^2 x$
2. Fractional identities
   - $\tan x = \frac{\sin x}{\cos x}$
   - $\cot x = \frac{\cos x}{\sin x}$

```
\displaystyle \cot{x} = \frac{\cos{x}}
```

Figure 7.8: Transferring a formula of goniometry from PlanetMath (on the right) to ActiveMath (on the left) using the smart-paste in jEditOQMath (middle).

Another common cause of error is the use of subscripts: they should, generally, be interpreted as variable names but often fail to be converted such. It is not yet clear at which stage such a conversion should be included, especially since some index notations are defined and actually converted: for example, x_A means the first coordinate of the point A. The right strategy could be to fallback on *escaping* variable names in case the conversion to content fails. Note, however, that within the current QMath the notation context should be enriched to process the indexed variable name as such.

All these failures are somewhat visible to the author and adjusting things is possible, from the easy crafting till the elaborate adjustments:

The first and basic adaptation that an author can use is to rely on the fact that the smart paste is separated from paste. A natural debug action is, thus, to first paste, inspect and arrange the text, then cut and smart paste. This is a manual process which is not particularly comfortable but works in many situations. It can enter a batch operation where all formulæ of a page are first *straight-pasted*, then a search and replace or batch manipulation is operated, then each formula is *cut* then *smart-pasted*. At each of these stages, the author is expected to maintain a readable source even if that source is not yet perfect for ActiveMath consumption.

The author has the option of adapting his environment for the smart paste function to operate with more success and quality.

The smart paste result may be guilty of dropping some symbols which is generally due to missing symbols in the notations context. The author, who is responsible for enriching it, can do so relatively easily, adding a new notation for the OpenMath* content he expects to receive. This enrichment task is explained at `http://eds.activemath.org/en/node/137`.

Clearly, the OpenMath* to QMath conversion could be more explicit, for example indicating that a symbol is missing. Unfortunately, there is no *warning* mechanism available yet in smart paste: it would involve a callback of the generated XSLT* transformation into the smart paste process. Such warnings should be detailed enough to actually add the notation if the author agrees.

Another adaptation to the conversion process from OpenMath to QMath could be the change of order of notations so that the right QMath notation is used for a given OpenMath. With QMath having been originally designed as a processor from a compact syntax to OpenMath, this has never been an issue until now. An example conversion is `a minus b` instead of the much more natural `a-b`. Thus far there has not been a *smart ordering* of the notations in the context except preferring shorter or binary operators, so that the author can change the notation context's order to indicate his preference.

A last dimension of adaptation is the rich domain of the Wiris MathML*-presentation to OpenMath converter. As described in [MEC$^+$06], this can be easily done using a desktop application called the *domain editor*. Thus far, however, it would require authors to install their own converter service and configure it which is really a developer task. We can envisage, however, that one day the domain is pushed from the client to the server. This approach is particularly relevant in complement to the WebEQ conversion which is non-extensible and unable to read many localized notations, such as the simple French $\mathrm{pgcd}(p, q)$ (meaning $\mathrm{gcd}(p, q)$) or the Russian and French notation of the binomial coefficient C_a^b denoting $\binom{a}{b}$.

7.6.6 Using the Smart Paste Function to Learn to Input

It should be noted that the smart-paste paradigm is considerably different than a batch conversion process; the small steps of the conversion and the ability to fix are two major differences. A most important one is also that it produces readable input that looks like the user had typed it by hand.

This ability allows a user knowing, for example, the LaTeX syntax, to discover ways to write QMath expressions by simply inputting desired things first in LaTeX then seeing its result in QMath. We expect this to become one of the major usages of the smart paste function.

7.6.7 Smart Paste Beyond Mathematical Formulæ

The same paradigm could be used in many other situations where the construction of the OMDoc encoding may require thought.

One particular aspect, which authors seem to have little interest in learning in detail, is that of multimedia embedding, such as Java* or Flash* applet* s. Their HTML* rendering has always been the result of an authoring-tool output, one that authors barely look at except for tiny aspects which they regularly control such as the size or *file-name*. The smart-paste function of jEditOQMath supports HTML* tags for applet* and Flash embeddings, inserting the necessary `omlet` and `private` elements of OMDoc. But the tag itself is not enough and the author still has to adjust the paths and/or copy the relevant files.

A natural wish would be to extend the smart paste function to larger paragraphs. However, this is much more difficult for the following reasons:

- Using the plain-text content that browsers put in the clipboard, there's almost no way to recognize formulæ islands. We shall need to get the source content in a different media-type* such as HTML* or RTF.
- The smart paste approach relies on a small number of pipelines that the user can easily differentiate by looking at their results. Presenting alternatives of complete paragraphs would probably make it much harder for the user to discriminate.

As a result, the current recommendation is to use smart-paste on individual formulæ or fragments of interest.

7.7 Conclusions

In this chapter, we have described the facilities based on the classical user-initiated transfer facilities – drag-and-drop and copy-and-paste – which support the learners and authors to input mathematical formulæ.

Two major conclusions have emerged from the web-based transfer capability offered to the learners from the content presentation to the Wiris Input Editor:

- For various technical reasons which are related to the original web-nature of the environment, the current transfer mechanism is based on the drag-and-drop of links. The students that were polled showed in majority that this transfer mechanism is unintuitive and too complex to be used regularly. What is needed is the application of the regular copy-and-paste paradigm.
- Textual input remains a most important ingredient of normal input methods of formulæ and this should be better supported including detailed feedback on errors.

We conclude with a sketch of research questions opened by the research reported about in this chapter.

7.7.1 Future works

Standards Based Copy and Paste Since the drag-and-drop implementation of LEACTIVEMATH, multiple discussions have taken place about the standardization of an API* to allow web-pages to inject or exploit supplementary content within the copy-and-paste paradigm. At this point of writing, three tracks appear to support this: HTML5, the web-apps working group's clipboard operations, and MathML* 3. It is not clear yet if one of these standards will be widely implemented in a way that allows mathematical formulæ to be copy-and-pasted easily following the regular paradigm. We expect the biggest stepping-stone to be the recognition of the safety of mathematical expressions within any encoding, maybe sanitized so as not to contain unsafe content (external references, scripting inclusions). Browser vendors, thus far, have only allowed content fragments which are fully trusted, moreover, special treatment has been made for the HTML* flavour which is expected to be very useful but needs sanitization.

Such standards would support putting multiple content formats into the clipboard. This may give a much finer control over the choice of involved formats and could open the door to the ability of selecting an expression in the simple view and obtaining the underlying mathematical representation; currently, all browsers we have met with all possibilities exploited invariably copy the plain-text conversion of the content. This makes, for example, $1 - \tan^2 x$ become `1-tan2x` where no converter can act: only the availability of MathML*, the underlying TeX* representation, or the semantic alternate representations can restore the meaning of such a formula which would be a natural candidate to be copy and paste into, for example, a spreadsheet cell function.

The implementation within web-browsers of the clipboard-specific aspects of MathML* 3 are the object of two tests within the MathML* test-suite, whose results for each contemporary browser are displayed at:`http://w3.org/Math/testsuite/`.

Exploiting the Clipboard-operations or HTML5 standards for such a server tool as ActiveMath would require engineering within the ActiveMath presentation pipeline coupled with JavaScript*. It may be stopped by selections which are not well-formed (such as $a + b-$ in $a + b - c$.

What is Mathematical Selection? Among the assumptions we have made in the learner oriented transfer function is that the rendering of the smallest sub-term under the mouse corresponds to the rendering of the smallest OpenMath* sub-tree of the rendered symbol under the mouse. This assumption is wrong in such expressions as $a + b + c$ which many tools encode in OpenMath as $(a + b) + c$: the smallest sub-term containing the first $+$ sign of $a + b + c$ is $a + b$ and the smallest containing the second $+$-sign of $a + b + c$ is $b + c$. What should be selected when the mouse advances over b then over $+$ to the right? The current rendering mechanisms of MathML*-presentation all apply a pure text selection mechanism, only selecting $b+$ while it would be natural to expect semantic services on formulæ that are pasted. Should the user be made aware that his selection is not well formed hence that OpenMath cannot be extracted? Should his selection be automatically expanded? These questions are even more delicate in a Web environment where common wisdom says that the selection that the user normally makes is something sacred that a web-page implementor should not be touching.[14]. A potential response to this problem might be that the selection should follow sub-trees of the MathML*-markup as well as all sub-trees in parallel markups (see [CIM10, Section 5.4]) but this has never been implemented as a selection mechanism.

Smart Paste as an Embedding Mechanism? Similarly to the original drag-and-drop actions of jEditOQMath, which allows references to content-items in an ActiveMath browser view to be drag-and-dropped to create *the right*`ref` element (see Chapter 3.4.2), smart-paste could be leveraged as a paradigm to include content from external libraries. Transferring URLs that follow a particular scheme (for example URLs of platform resources see http://i2geo.net/) could apply an amount of heuristic to the following: embed the necessary player code, connected to the server or importing its content, include a title and copyright information from the resource and include other metadata* information such as the trained topics, provided a mapping can be made between the ontology of annotations on the external library and the ontology of target concepts in the current ActiveMath collection.

7.7.2 No Palettes, just Samples

As a last contribution to this chapter, we propose the outline of an ideal system for formula input in web-browsers following the evaluation results. It should allow direct text input into a normal text box embedded in the webpage that directly renders the formula in the graphical format, possibly in an embedded Java* applet* below the text box. The applet* should not distract from the exercise, or other task, in any way and must be embedded in the same page. Drag-and-drop or copy-and-paste should be permitted into the text box with immediate rendering below and the rendering could also highlight syntax errors. These errors must have detailed feedback and suggested corrections.

The integration of buttons (that is, of notations templates) with such a system is tricky as it might steal screen space from the main window, thus obtaining the same isolation as the current Wiris Input Editor. One solution could be a pop-up menu with subcategories similar to those already used in the current input editor. This could be navigated directly from the text box and overlaid on the

[14]A recent thread on the W3C's Technical Architecture Group's public mailing-list was started by Tim Berners-Lee echoing his dismay when encountering that a selected text he copied was enriched by undesirable tracker URLs. This thread shows the visceral reaction to the empowerment of web-page authors to interact in the clipboard. See this thread named *Copy to Clipboard - ambush and abuse by javascript* at `http://www.w3.org/mid/AFFAB130-B693-4AC9-91E6-B6834E57B3F5@w3.org`.

main window which does not distract as it would only be visible when the user calls for it.

An alternative way to achieve the same notation template function of the input editor with such an approach would be to rely more on the transfer mechanism, possibly making such a notation template's buttons appear more like a *book of notations* which should be contributed to by the learning content being browsed, and by domains already discovered. This book of notations would be maintained by the learners.

Such a system would probably avoid the spread of the dominant opinion that an input-editor is too complex for small tasks, but would provide the same direct-action immediacy of said input-editor. It could avoid slow error reporting cycles by the immediate display of a partial formula as well as the error-highlight in the expression, one of the main critiques to the text-input. The simplicity of the plain-text nature, both being input by the user (when typing) and by the computer (when dropping or pasting) would be honoured, an important aspect as noted in this evaluation.

Chapter 8

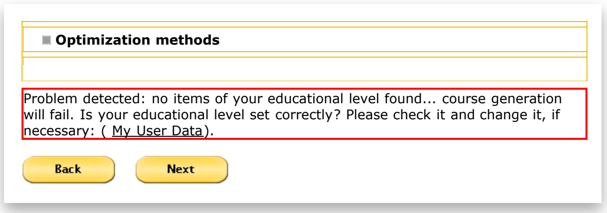
Helping the Author to Input Knowledge for the Modules of ActiveMath

8.1 Introduction

This chapter addresses the input of all information in the content that *represents* the knowledge necessary for the intelligent features of ActiveMath. This knowledge is encoded in the properties of content items and the relationships between them within the `metadata` and `extradata` children.

As described in Chapter 2, the following components of the ActiveMath learning environment are impacted by the **metadata**:

- The presentation system uses the metadata in a straightforward manner: simply modifying the display of the content item depending on some metadata. For example, the difficulty of an exercise is often shown with a number of stars; the item type (and sub-type for `omtext` elements) is shown as an icon next to of the title.
- The assembly tool acts in a similar manner.
- The exercise system uses the metadata so as to broadcast it to other components, as part of an event that describes the attainment of a step in an exercise. The metadata of the exercise, or of a step, is sent so that the user-model updates its beliefs.
- The user-model uses the item-types, the relationships, and other properties to create its graph of beliefs and exploit the events sent by the exercise system to update itself.


Optimization methods

Problem detected: no items of your educational level found... course generation will fail. Is your educational level set correctly? Please check it and change it, if necessary: (My User Data).

Figure 8.1: Trying to obtain a generated course on optimization with the user-data of a student in secondary school lead to this message: no suitable content.

- The tutorial component uses many of the metadata properties of the content items to select the items according to its scenarios.

Hence, an author that wishes to use these features should input the right metadata. As explained in Chapter 9, explanations about the meaning of the knowledge represented by the metadata elements have been given by the presentation of a *metadata specification*, a simple reference text describing the meaning of each of the properties. This practice has proved to be insufficient in most cases and, generally, content collections became mature with the usage of the artificial intelligence features only with a very intensive maturation phase where tests occurred intensively.

This chapter proposes a method to organize the metadata elements so that it is considerably easier to manage. Together with the short reload cycles (see Section 3.4.3) we claim that this method makes it easier to provide the metadata necessary for the intelligent features to run and that it is, thus, easier to attempt.

8.1.1 Outline

We first describe the issues relevant to the management of metadata that we experienced when in contact with the authors. We then describe the proposed paradigm, metadata inheritance, and detail how it is encoded in the OMDoc elements. A description of how the inheritance is implemented in the content storage follows. Related works are then described, followed by open questions.

8.2 Issues with Metadata Management

The first and foremost experience that authors encounter when attempting the course generation in the default collections shipped with the traditional jEdit-OQMath was that the course generation fails with a message indicating that there is no content ready for your learning context, as displayed in Figure 8.1.

Certainly, an experienced author would know how to solve it: the content needs to be enriched with the given `learningcontext` property, or the user profile's data needs to be adjusted. This issue can be easily overcome: until our proposed contribution, one simply had to insert *the right learning context elements* for each of the content items; in our experience, this made the metadata elements of content items very verbose. Many other adjustments and trials are needed to make sure that the course generation, in all of its flexibility, gives pedagogically sensible results.

The meaning of each of the metadata field can be read from the metadata specification, but this does not help to grasp fully what is involved for the course generation or for the user-model. To this issue, one can easily respond that an explanation of all the AI details behind the learner model or behind the course generation is inappropriate for normal authors. What remains, beyond the simple *meaning* is to grasp the implications by experience. Example of such experience is the effect of a successfully solved exercise on the user-model: it is clear that the dependencies are followed, but *how much*? An author would wish to attempt it. Such attempts remain quite elaborate (one has to input the right metadata, then load the content, log-in with *correct user*, play the content (which includes solving the exercise, right or wrong), and view the result. In this workflow, any possible simplification is welcome.

The issues described above are of an experimental nature and such experience may come much later than at a testing stage in the middle of authoring. It may well be that a content collection, for example, also becomes interesting for a new field of study. Thus, changes to the metadata may be necessary long after the first authoring experiences, and finding the exact place to adjust may be a daunting task; moreover, changing at many places, for example operating with a global search and replace, is an error-prone task. Changes should be minimized as much as possible.

8.3 Metadata Inheritance for ActiveMath

In Chapter 4, we have described an important authoring artifact for an easier management of references within the content. In this chapter we describe a facility to make it easier to manage another part of the content which may take up a lot of space while not being explicitly visible: the metadata. In its general form, metadata is simply the set of data that is *aside* of the content. Most of the time, metadata includes bibliographical elements – OMDoc uses the widespread standard Dublin-Core* to this effect. In ActiveMath, metadata encompasses multiple elements that describe the properties of the content item under a pedagogical aspect. Most of these elements are children of the extradata element and used by ActiveMath's components for services to the learners.

In many cases, metadata is the same for many content items in a collection or in a document. An example: the author of each item in a document may well be the author of the document; another example: the collection has been designed for students of electrical engineering in their 1^{st} year of university.

Metadata inheritance can be used to this effect: it is used in OMDoc (see [Koh06, Section 12.4]) which stipulates that the bibliographical elements missing inside a metadata element should be copied from the enclosing elements' metadata. In this chapter, we propose a generalization. The pattern described here is the practice of avoiding the repetition of metadata fields as long as they can be inherited from an enclosing element.

Metadata inheritance can be applied in three different approaches. For each 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. The three approaches are:

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

It should be noted that metadata inheritance works on properties and not on element names. Metadata properties are made of the element-name and all the attribute names and values of the element. For example the property of a russian translator name may be inherited and not conflict with the property of a French translator name, even if stipulated at another metadata location.

ActiveMath's metadata inheritance differs from OMDoc's in several respects:

First, it allows the content items to inherit the metadata from the collection-descriptor which is a metadata record of the whole collection (an OMDoc `metadata` element with the id `_collection_metadata_`).

Secondly, it offers metadata inheritance by reference to indicate the intent of inheriting metadata from an item at a completely different location. A potential usage of inheritance by reference 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 an element child of the `metadata` element in the form `<inherit from="pointer"/>` and should be interpreted as the metadata of the closest enclosing element (ignoring inherited metadata of the referenced item, thus avoiding cyclicity).

Thirdly, ActiveMath's metadata inheritance is configurable: all children of the `metadata` or `extradata` elements can have the `inherit` attribute with one of the above values which describe the inheritance types. This gives flexibility in the inheritance model, allowing it to stop the inheritance at parts or even to group items that share properties within one document, entering only the common information on top of it.

Finally, ActiveMath's metadata inheritance applies to the broad spectrum of the metadata properties of ActiveMath which extends OMDoc metadata with pedagogical and mathematical properties. For each of them, the default inheritance policy is encoded in the DTD* that is shipped with ActiveMath and with every collection created by jEditOQMath. It 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` as specified in OMDoc 1.2
- `domain_prerequisite`, `prerequsite`, `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`

The rules above describe a relatively simple model of inheritance which could be reasonably understood by the authors. But, because offering metadata inheritance implies spreading metadata into several places, it may be necessary for the authors to preview the result of the metadata inheritance. We describe the details of this feature below.

The reader may be surprised to see that this inheritance model is coming up with such the question *Does it make sense to inherit misconceptions?* The answer to this question is probably "no" but it could be "yes" if someone actually inserted the misconception metadata property on top of a file: such an insertion would make no sense otherwise.

8.4 Implementation of Metadata Inheritance

Following the content storage loading process described in Chapter 8, the metadata inheritance is operated as a part of *decontextualization*: when pulling the item out of context, it needs to carry inside it all the information that was part of its context. It is, thus, natural that the metadata inheritance is a service of the content storage alone, and that it is ignored by other components of the ActiveMath learning environment.

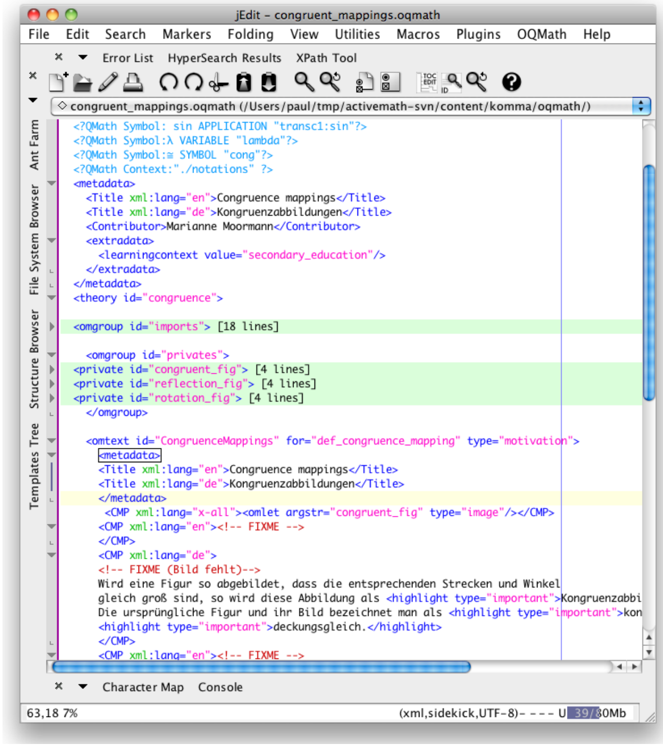
Metadata inheritance works with SLuMB, the contemporary content storage described in Section 4.4.4. It is implemented as follows:

- within the scanning phase, the content items identifiers and the `inherit-from` elements are indexed
- following this, the elements that are targeted by the `inherit-from` elements are loaded in memory; similarly, the collection metadata is loaded in memory
- when loading a file, the metadata inheritance is applied to each content items from these memory-stored elements as well as any parent through a simple JDOM* manipulation. Because the default inheritance policy is indicated in the DTD*, it is present in the parsed elements and the configurable policy can be followed
- the indexed streams of the content items and their metadata are done with all inherited values; a content delivery following an MBaseRef query will thus respond the content item with all inherited values.

This implementation allows all of the inheritance flexibility described above. It is in active use. It can be made within a reload cycle as well, in this case, all available information in files that are unchanged are pulled from the index.

This implementation also allows the author to get a **preview of the metadata inheritance**: within jEditOQMath, having positioned his cursor within the content item of interest, he invokes the metadata inheritance preview command from the OQMath menu. This results in the content item being enriched with all the metadata information which are effectively previews. Typically, the author will review this result and cancel the last action since he is probably not interested In keeping duplicated inherited values, but he will know what is in the context. A screenshot of a content item source before and after metadata inheritance is in Figure 8.2: one sees that the `Contributor` and `learningcontext` elements are inherited into this `omtext`.

The metadata inheritance preview is implemented in almost the same way as the *Generate Imports** command which is implemented as a special form of the reference resolution process (see Section 4.7.3). The inheritance preview performs all the tasks but does not store the result; instead it returns it to the calling jEditOQMath.

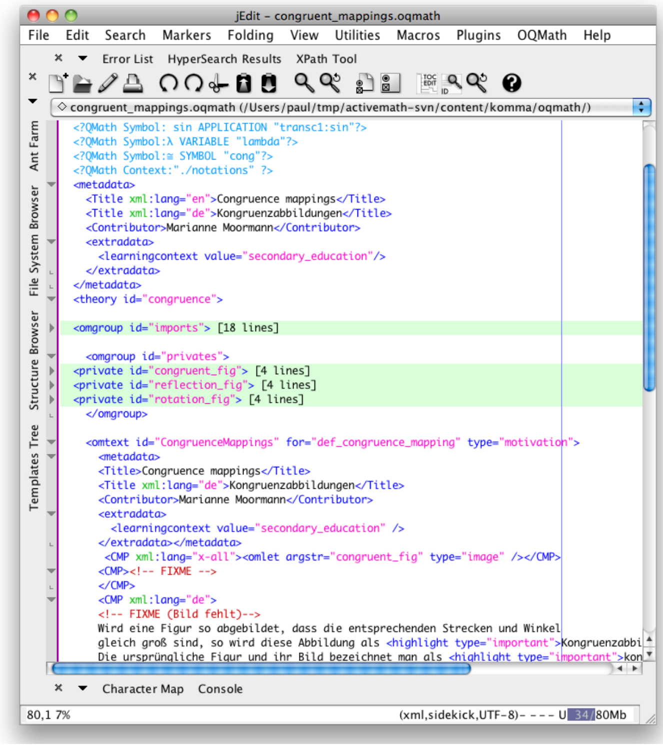

jEdit – congruent_mappings.oqmath
File Edit Search Markers Folding View Utilities Macros Plugins OQMath Help
Error List HyperSearch Results XPath Tool
congruent_mappings.oqmath (/Users/paul/tmp/activemath-svn/content/komma/oqmath/)
Ant Farm File System Browser Structure Browser Templates Tree
Character Map Console
63,18 7% (xml,sidekick,UTF-8)- - - - U 39/80Mb

```
<?QMath Symbol: sin APPLICATION "transc1:sin"?>
<?QMath Symbol:λ VARIABLE "lambda"?>
<?QMath Symbol:≅ SYMBOL "cong"?>
<?QMath Context:"./notations" ?>
<metadata>
  <Title xml:lang="en">Congruence mappings</Title>
  <Title xml:lang="de">Kongruenzabbildungen</Title>
  <Contributor>Marianne Moormann</Contributor>
  <extradata>
    <learningcontext value="secondary_education"/>
  </extradata>
</metadata>
<theory id="congruence">

<omgroup id="imports"> [18 lines]

  <omgroup id="privates">
<private id="congruent_fig"> [4 lines]
<private id="reflection_fig"> [4 lines]
<private id="rotation_fig"> [4 lines]
  </omgroup>

  <omtext id="CongruenceMappings" for="def_congruence_mapping" type="motivation">
    <metadata>
    <Title xml:lang="en">Congruence mappings</Title>
    <Title xml:lang="de">Kongruenzabbildungen</Title>
    </metadata>
     <CMP xml:lang="x-all"><omlet argstr="congruent_fig" type="image"/></CMP>
    <CMP xml:lang="en"><!-- FIXME -->
    </CMP>
    <CMP xml:lang="de">
    <!-- FIXME (Bild fehlt)-->
    Wird eine Figur so abgebildet, dass die entsprechenden Strecken und Winkel
    gleich groß sind, so wird diese Abbildung als <highlight type="important">Kongruenzabbi
    Die ursprüngliche Figur und ihr Bild bezeichnet man als <highlight type="important">kon
    <highlight type="important">deckungsgleich.</highlight>
    </CMP>
    <CMP xml:lang="en"><!-- FIXME -->
```


jEdit – congruent_mappings.oqmath
File Edit Search Markers Folding View Utilities Macros Plugins OQMath Help
Error List HyperSearch Results XPath Tool
congruent_mappings.oqmath (/Users/paul/tmp/activemath-svn/content/komma/oqmath/)
Ant Farm File System Browser Structure Browser Templates Tree
Character Map Console
80,1 7% (xml,sidekick,UTF-8)- - - - U 34/80Mb

```
<?QMath Symbol: sin APPLICATION "transc1:sin"?>
<?QMath Symbol:λ VARIABLE "lambda"?>
<?QMath Symbol:≅ SYMBOL "cong"?>
<?QMath Context:"./notations" ?>
<metadata>
  <Title xml:lang="en">Congruence mappings</Title>
  <Title xml:lang="de">Kongruenzabbildungen</Title>
  <Contributor>Marianne Moormann</Contributor>
  <extradata>
    <learningcontext value="secondary_education"/>
  </extradata>
</metadata>
<theory id="congruence">

<omgroup id="imports"> [18 lines]

  <omgroup id="privates">
<private id="congruent_fig"> [4 lines]
<private id="reflection_fig"> [4 lines]
<private id="rotation_fig"> [4 lines]
  </omgroup>

  <omtext id="CongruenceMappings" for="def_congruence_mapping" type="motivation">
    <metadata>
    <Title>Congruence mappings</Title>
    <Title xml:lang="de">Kongruenzabbildungen</Title>
    <Contributor>Marianne Moormann</Contributor>
    <extradata>
      <learningcontext value="secondary_education" />
    </extradata></metadata>
     <CMP xml:lang="x-all"><omlet argstr="congruent_fig" type="image" /></CMP>
    <CMP><!-- FIXME -->
    </CMP>
    <CMP xml:lang="de">
    <!-- FIXME (Bild fehlt)-->
    Wird eine Figur so abgebildet, dass die entsprechenden Strecken und Winkel
    gleich groß sind, so wird diese Abbildung als <highlight type="important">Kongruenzabbi
    Die ursprüngliche Figur und ihr Bild bezeichnet man als <highlight type="important">kon
```

Figure 8.2: Metadata inheritance preview.

8.5 Alternative Approaches

The Trial Solution's Small Learning Objects Within the Trial Solution project (2000-2003) and the In2Math project (2001-2004), the concept of Slicing Books Technologies was further developed. It shares a vision with ActiveMath in terms of content-selection mechanism: offer books to learners that only contain relevant material to the learner's objectives and knowledge. Among others, the paper [Dah06] describes lessons learned from this technology. The objective of slicing the book in to many small pieces is supported by metadata annotations on the sliced units, including dependencies between them. The approach of inheriting metadata described there is a simple top to bottom inheritance along the *default organization* of the *content package* (an IMS CP package), which represents a *sliced book*.

The small learning objects approach of [Dah06] is somewhat close to ours, but our approach is more flexible due to the fact that inheritance is enabled by default and that it can be pulled from an outer element. This approach does not support re-using someone else's content. While our approach could allow mixes of books that are not well controlled, i.e. that do not make sense pedagically, enabling re-use remains a first important step.

Metadata Generation Because assigning metadata is a difficult operation, several research teams have investigated *metadata generation* which involves producing many of the fields of the metadata record automatically.

Not surprisingly, the context in which the metadata record would be generated is stated in [CMD05] as quite important. An example context is the user-data with which the user contributes a learning-object within a repository: from there it is easy to suggest the author data and language; similarly, from the history of previous contributions, many other fields can be suggested.

Several other approaches employ content analysis to generate metadata, for example [RSG08].

While these investigations certainly help to provide the first input, they do not seem to be designed towards the authoring of evolving content. This thesis has pinpointed how important the multiple attempts and incremental adjustments are. Our approach is based on metadata encoded within the content and is thus part of these cycles, while an approach of posting to a repository and filling the metadata fields is likely to start it all over again. We believe that the author will want to modify the generated metadata; if re-generated at every revision, this modification should not be needed again. The insertion into the authoring workflow needs to be studied – metadata generation should probably rather be formulated as *metadata suggestion*: an approach where the training of the machine learning components that produce the metadata become retrained as the author accepts, discards, or corrects the suggested values of the metadata fields.

Scope of the Metadata Another aspect of metadata which the research literature has covered of is that of defining the scope of metadata. Metadata being data about data, can be quite arbitrary, but its scope can be made more precise by defining its expected applications.

Asked about the finality of metadata generation in a workshop close to the writing of [CMD05], Erik Duval answered that "it was not clear yet how to state the quality of a good metadata record for a learning object". The best answer was, thus far, that the learning object learning object would be found when the appropriate query was formulated within the repository. This fuzziness is probably the source of many issues relating to the interpretation of metadata.

In OMDoc and ActiveMath, metadata is part of the content, a well defined set of files; moreover, the objective of metadata is clear: it must play well in the ActiveMath environment under various features. Stating this precisely is the cornerstone of an efficient metadata input by authors who can work towards the objective. The statement we have often made about metadata being of a semantic nature, although not undesirable, is much less precise and does not guide the authors to an actual test.

In the situation where a repository of learning objects is shared in a community, the finality of metadata is quite different: it often is that the resource should appear under particular queries which would be typically expected of other users.

8.6 Conclusion

In this chapter we have presented a mechanism to simplify the management of the metadata in ActiveMath content items. This metadata represents the knowledge that is the essential input for much of the intelligent behaviour of the ActiveMath learning environment.

The approach can be summarized as *putting the right thing in the right place* as it allows an author to group content items which share common metadata properties and let the metadata property only be input once – for example on top of the root element of the document. Using this approach, the author can aim at an ideal layout of information where everything is in its place, *pauca sed structura* (poor but structured).

8.6.1 Open Questions

Experimental Validation Unfortunately we have no experimental measure of the practicality of the metadata inheritance. It was welcomed by our authors but we have not been able to measure whether it really allows, for example, an easier access to the usage of intelligent features for which we have seen multiple difficulties, as reported in Chapter 9.

The Danger of Living in Context The usage of imports and of the metadata inheritance, as well the inclusion within a collection, all allow content to be more economic, hence more readable; leaving the *well known attributes* to single places central to the content collection. Similarly to relative URL-references in HTML*, this has both the danger and advantage of changing context when the location is changed: a fragment of OMDoc copied and pasted to some other collection will keep its relative references as relative and will inherit from the new context. In some cases, this will be advantageous (when moving between similar collections for example, inheritance of an exercise's learning-field because it is moved to the file of exercises in a new learning field); in other cases it may cause some surprise: inheritance being unexpected and references being wrong or broken. To protect against such surprises, the reference validation is fundamental (it is highly probable that inappropriate relative references will actually become broken references), and therefore the metadata inheritance preview can be used. We expect these protection measures to be sufficient.

Is Inheritance flexible enough? ActiveMath's metadata inheritance is configurable *top-to-bottom*, it remains to be validated experimentally that this inheritance is sufficiently flexible. We anticipate that the inheritance to be configured during times of discussions between content development experts and regular author teams – for example during interactions during a research project that involves content development. We envision a project to work on multiple educational fields because they aim at entry level university courses while it will be also common that field-specific content is written. Will there be projects that are moving things around in a way that inheritance becomes a burden? Only long-term experience will tell.

Enhancing the Test of Intelligent Features As we have indicated, a major difficulty in authoring the metadata lies in the understanding and testing of this input: much of this metadata only has an effect if part of a complex process (such as the user-model update or course generation). Enhancing the testing of this metadata is thus an essential future task.

A first stab at this testing is in ActiveMath's Gap-detection system [Jed10]. This, basically, applies patterns of typical metadata provision and reports the misses. Unfortunately, this architecture is not yet integrated within the jEditOQMath-based authoring workflow.

Multiple other means could be investigated:

- To make it possible for the states of the learning environment to be reproducible by a simple browser reload (e.g. user-model preview, an exercise run, or the re-delivery of an exercise).
- to share the gap reports as a specification of the objectives of further development of the content collection.

- To make more widespread the practice of *demo scripts* so that it is easy to share the experiences with the learning environment; for example, it should be easy for a pedagogy expert to describe an intended scenario and an encoder to encode the content for it (as has been done for the Mathe-führerschein project).
- To advertise content quality standards which include the pedagogically founded usability of the resulting learning experience.

It is our belief that enhancing the testing practice is one of the major research areas that authoring tools research should take, so as to support the continuous evolution of intelligent learning tools.

Chapter 9

Case Studies

The previous chapters have described several technical solutions to help the authoring process. These solutions have emerged from the authoring practice that took place during the projects that supported the development of the ActiveMath learning environment.

In this chapter, I provide a more precise description of the approach we have taken to advise authors and to support them to write content. We also study the cases of a few representative authors and describe the knowledge gathered by their experience.

9.1 Experimental Approach

This thesis is the result of a permanent *side activity* which has never been the central focus of any research project thus far. In each project (VIL (2000-2001), In2Math (2001-2004), MMISS (2001-2004), LeActiveMath (2004-2006), Mathe-führerschein (2004-2006), MatheBrücke (2007-2011), ActiveMath-EU (2006-2008), ALOE (2008-2010), ATUF (2007-2009), AMor (2008-2010), eCel (2008-2010), Math-Bridge (2009-2012)), an amount of learning resources was to be developed or refined for the ActiveMath learning environment. Only two of them had authoring tools in their development plans and even there as a minor scope: MMISS and Math-Bridge.

Authoring activities were performed by previously identified educators with broad teaching experience using available tools. As I described in Chapter 3, authoring source OMDoc content has always been discussed and envisioned as a possibility and this is what we report about.

The responsibility to develop authoring tools has often focussed on a few authors at a time, and has centered around advising on the available tools necessary to evolve a given learning experience so that it can be presented in ActiveMath. This would run parallel to developing further tools to help the authoring work. Most of the time, my role has been to mediate between an author's expectations of the learning environment and what was achievable in realistic timelines (sometimes including ongoing software development).

Compared to classical design and evaluation methodologies, our experimental approach is rather wild and uncontrolled. Indeed, no formal hypothesis verification could be realized. Examples of formal evaluations include that of MOT in [CSBC07], of the Assistments' Builder in [RPA$^+$09], and of CTAT in [AMSK06]. Such evaluations are generally realized in short time spans, often with *students at hand*, and generally with no possibility to reach a routine usage of the authoring tools and the learning platform. Much of the instrumental genesis process where authors reach an efficient usage of the tools, accepting their limitations, is lost.

One report on the effectiveness of authoring tools differs from this: the study of REDEEM's usage and learning effectiveness in [AMG$^+$03]. This usage report is presented to support the description of the authoring tools; it presents a qualitative analysis of the lessons learned. We follow this pattern in more detail, by presenting the instruments used to introduce and support the authoring practice and by describing typical authors; lessons learned from these experiments are then described.

9.2 Support Instruments

9.2.1 Initial Tutorials

Persons interested in learning how to use ActiveMath and authoring tools to create content have been invited to so-called authoring tutorials, sometimes also called *trainings* or *workshops*. These tutorials were mostly organized as in-presence sessions. They consisted of the following steps:

- installation of the learning environment and authoring tools, basic functions proofing (this varied in duration from 15 minutes till 2 days)
- demonstration of the system
- conceptual introduction to the content's knowledge representation
- practical introduction to editing simple content items
- presentation of participants' objectives
- practical introduction to authoring (semantic) mathematical formulæ
- practical demonstrations following participants' requests

The tutorial was meant to endow the participants with basic abilities to author, expecting them to benefit from continuous community oriented support later on. The results varied strongly from one author to another, depending on the maturity of the tools at the time, the capacities of the laptop, or the familiarity with the environment. It generally lasted two to three days.

Traces of some of these tutorials are found at `http://eds.activemath.org/en/taxonomy/term/40` as well as at `http://www.activemath.org/projects/authoring`.

The tutorial was once realized as a series of five online sessions. A similar configuration as above was deployed (each author had his own machine where both communication and attempts were done) supplemented by a screen broadcast function (through Darwin Streaming Server) and a chat space (through Skype). For these, a considerably stronger set of instructions were deployed in order to make sure the installations were done and a previous individual check service was offered. Many traces are left of this tutorial at `http://eds.activemath.org/en/online-tutorial`. Although each participant had more time to try more things on his own, the overall involvement was smaller, probably due to the remote nature.

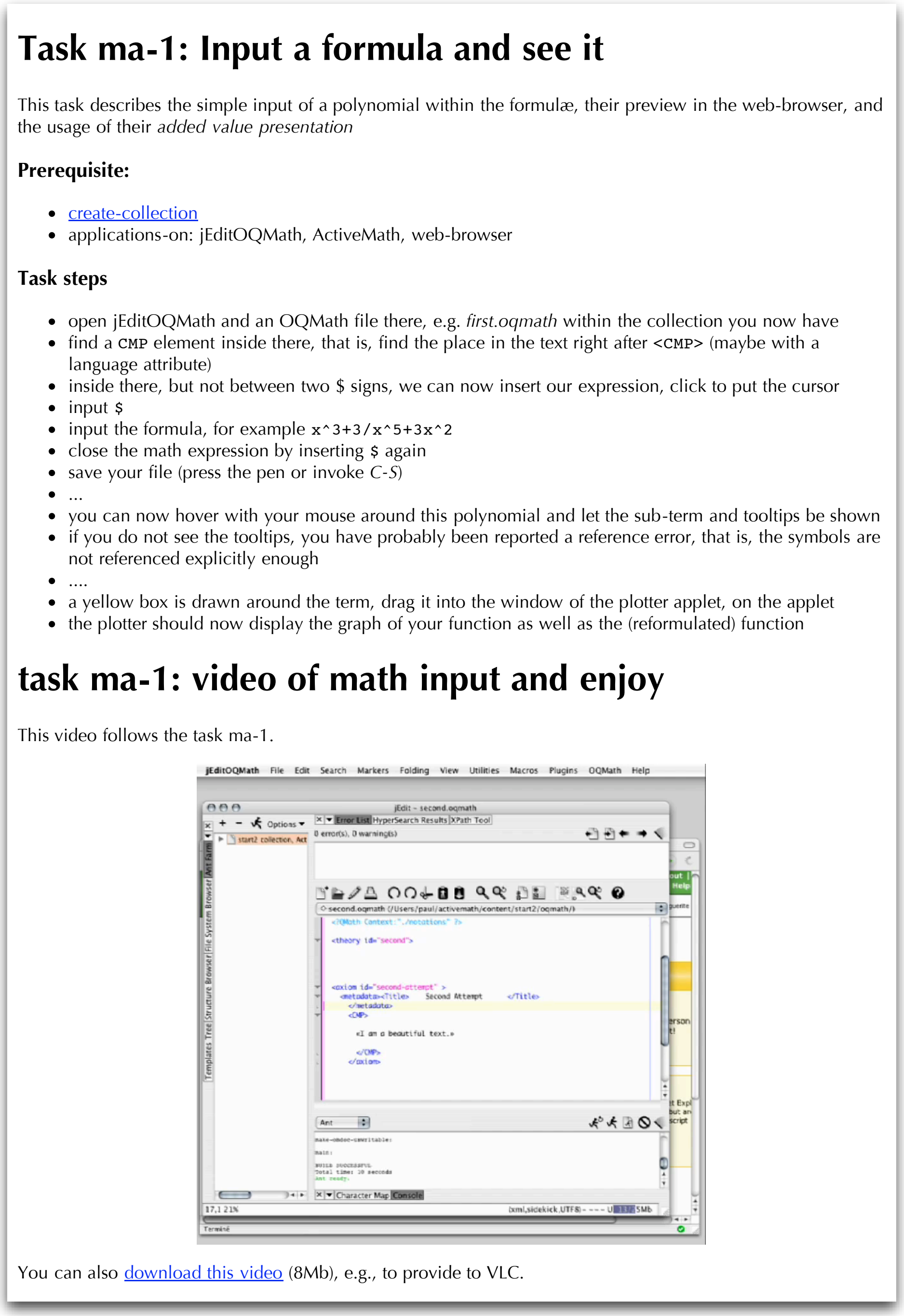
9.2.2 The Book of Tasks

Following a few authoring workshops, the need to keep track of the authoring steps appeared early. As commonly done in such tutorials, slides were presented and made available; but they did not describe a concrete sequence of actions. In the online tutorial, the video recordings were made available, but their duration (five times 5-10 hours) and the lack of tagging made them hard to use as a way to reference back to the actions that were learned or demonstrated.

What crystallized is the understanding that *solutions* of exercises distributed in the tutorial sessions were a good source of know-how to create content. Strongly encouraged by one of the most eager learning authors, Albert Bäumel, sequences of very concrete steps described as recipes were written in an online book called the *book of tasks* that can be browsed at `http://eds.activemath.org/en/tasks`. Each of the tasks are described with prerequisites (which may include installed and running systems), achievements, and the steps to perform them in all details. Sometimes, tasks are complemented by a screen-recording where the task is achieved. A sample task and its video are depicted in Figure 9.1.

9.2.3 Visible Issue Management

Authors are definitely among the most important users who can, through the content they provide, assemble learning experiences that invite the learners to use the environment so as to learn effectively. It is not surprising, thus, that authors have



Task ma-1: Input a formula and see it

This task describes the simple input of a polynomial within the formulæ, their preview in the web-browser, and the usage of their *added value presentation*

Prerequisite:

- create-collection
- applications-on: jEditOQMath, ActiveMath, web-browser

Task steps

- open jEditOQMath and an OQMath file there, e.g. *first.oqmath* within the collection you now have
- find a `CMP` element inside there, that is, find the place in the text right after `<CMP>` (maybe with a language attribute)
- inside there, but not between two $ signs, we can now insert our expression, click to put the cursor
- input `$`
- input the formula, for example `x^3+3/x^5+3x^2`
- close the math expression by inserting `$` again
- save your file (press the pen or invoke *C-S*)
- ...
- you can now hover with your mouse around this polynomial and let the sub-term and tooltips be shown
- if you do not see the tooltips, you have probably been reported a reference error, that is, the symbols are not referenced explicitly enough
- ....
- a yellow box is drawn around the term, drag it into the window of the plotter applet, on the applet
- the plotter should now display the graph of your function as well as the (reformulated) function

task ma-1: video of math input and enjoy

This video follows the task ma-1.

jEditOQMath File Edit Search Markers Folding View Utilities Macros Plugins OQMath Help
jEdit - second.oqmath
Options
start2 collection, Act
Templates Tree | Structure Browser | File System Browser | Ant Farm
Error List HyperSearch Results XPath Tool
Ant
Character Map Console
Terminé

0 error(s), 0 warning(s)

second.oqmath (/Users/paul/activemath/content/start2/oqmath/)

```
<?QMath Context:"./notations" ?>

<theory id="second">



<axiom id="second-attempt" >
  <metadata><Title>    Second Attempt      </Title>
    </metadata>
    <CMP>

      «I am a beautiful text.»

      </CMP>
    </axiom>
```

```
make-omdoc-unwritable:

main:

BUILD SUCCESSFUL
Total time: 10 seconds
Ant ready.
```

17,1 21% (xml,sidekick,UTF8) - - - - U 5Mb

You can also download this video (8Mb), e.g., to provide to VLC.

Figure 9.1: Extract of the task `ma-1` describing a first experience in inserting formulæ found at `http://eds.activemath.org/en/node/200`.

been among the users with the most active issue reporting: within the projects, their expectations of new or enhanced features, their notes of bugs, and the minimum requirements they set for learning are all registered as issues. They are important traces of the didactical sense they made of their attempts to use ActiveMath.

Quite often, such issues have been filed following support requests, because they could not be solved by providing a simple explanation of *how to do it*. Their concretization as an *issue* with a fixed URL allows the evolution to be tracked and, thus, authors to be notified when it is fulfilled for them. Issues of the ActiveMath platform are displayed at `http://jira.activemath.org/browse/AMATH` and authoring issues are at `http://jira.activemath.org/browse/AMAUTHOR`.

9.2.4 Authoring Communities

As described in Chapter 6, best practice of authoring includes the disposition of authoring sources in *projects* which are shared by communities of authors all looking at the same project under their perspective.

This sharing practice has been honored in all content development projects and is fairly natural. However, it is not sufficient alone to create a community of authors long term. Requirements for such communities have been outlined in Chapter 6.

A first approach was realized using Drupal, a widespread community server offering wiki, forums, and many other community tools: `http://eds.activemath.org`. Only its wiki nature stayed in use. The ideal situation has been attempted with the usage of the LibreSource web-platform and so6 versioning system [SMMMG06] which supported all the necessary features (easy versioning, distributed versioning, web-visibility of actions, integrated timeline view of all actions relevant to the project, ...). A few examples are still visible online, among others the Combien-collection, realized in the ActiveMath-EU project (`http://www.activemath.org/projects/CombienCollAM`).

Unfortunately, the so6 versioning solution of LibreSource turned out to be not sufficiently sturdy and further activities have reverted to the usage of Subversion.

Online authoring communities have been a central point of contact in regards to the authoring activity. The availability of sources allowed one to see how far the authoring teams were, and to provide support on concrete situations which all the community can see. In turn, such activity can start to become a source of good know-how built collectively.

9.2.5 Remote Support

I have indicated above that much of the authoring development activity was spent in advising the authors on best usage. After the initial tutorials, such advice has mostly been dispensed via email with a few sessions of instant communication for deeper understanding when problems of misunderstandings crept up in email exchanges.

When all this support still was insufficient for me to understand a desire or problem, or for the author to realize fully his intents, *remote desktop* sessions were also organized.

9.3 Case Studies of Selected Authors

Overall, about 40 users of the ActiveMath learning environment also used jEdit-OQMath to create portions of content. From these, we have selected representative users who have achieved a significant amount of authoring work. We list them in chronological appearance of when they began contributing to ActiveMath authoring.

9.3.1 Barbara Grabowski

Barbara embarked on authoring in the early days of ActiveMath. She was the fourth author of OMDoc documents, after Michael Kohlhase, George Goguadze, and Jochen Büdenbender. She started to author content of statistics for her teaching in the Hochschule für Technick und Wirtschaft in Saarbrücken. She attended a few tutorial sessions and support sessions, and was aided in the ActiveMath deployment and customization at her high-school by a few assistants.

Barbara comes as a Professor of Mathematics, specialising in statistics, and had experience with multiple mathematical softwares including the R computer-algebra-system and the TeX* typesetting system.

She learned ActiveMath authoring during visits to the DFKI, and on her own. She frequently visited DFKI to discuss or obtain support.

In the In2Math project, the first collection she wrote was based on the *authoring kit*, in the QMath language (see Section 3.3) with which she wrote two chapters of statistics complemented by a few multiple-choice-questions, and computer-algebra-connected external exercises. The hard facets at this period were the following:

- QMath was in its youth and error reporting was relatively poor; moreover, the content-storage reported errors (such as reference errors) were not reported with locations (they were part of the log of the *buildIndex** command). Nonetheless, Barbara did create several symbols for statistics for which our team wrote the XSLT* extensions needed to render them in TeX* and HTML*.

- the ActiveMath server software was also young and no releases were provided yet; Barbara performed cvs-updates from time to time but such an operation always had the risk of bringing her authoring ActiveMath in to a fragile state. Conflicts (e.g. between HTW-special adaptations and the trunk), compilation errors, or broken functions are all issues that occurred and she was only able to fix these by requesting our help.

- She always acted *oriented to the final product*, accepting compromises in order to stay realistic and obtain a maximum of the desired features. For example, when formulæ weren't representable in OpenMath* or in QMath, she used a blend of text and formulæ.

- throughout her authoring life, she has been requesting reference documents of the elements that ActiveMath can make use of. This service has never been fully answered although partial answers were done later (e.g. the metadata* report). See Section 10.2.3.

Around 2004, her content collection was converted from QMath to OQMath but no project funded the development of content any longer. Barbara started more intensively interactive exercises for statistics and calculus using Prolog for which she programmed and authored.

In a later project, MatheBrücke, Barbara went back to authoring, together with employees Susanne Gäng and Melanie Kasper. They realized `mathebrHTW`, a collection of content for remedial courses for entering HTW. The result, consisting of about 70 pages of static content and about 100 interactive exercises, is being evaluated with learners as of this writing.

9.3.2 Christian Gross

Christian began authoring for ActiveMath within the LeActiveMath project in 2004, part of the team of the University of Augsburg, Department of Mathematics Pedagogy. His education includes a doctorate in Mathematics and his work was that of a pedagogy expert and content creator. In LeActiveMath, the team of Augsburg had the mission of creating content, advising on pedagogical strategies, and letting content be used in controlled experiments in colleges nearby.

Christian had experience with the usage of the web and with the usage of the TeX* composition system, which made the paradigm of source editing natural to him. He learned ActiveMath authoring in a live training session at the start of this project and by asking questions directly, often on the users mailing-list. The jEditOQMath he used had no smart-paste and no refactoring.

He has been writing a large part of the content of LeAM_calculus*, in German and English, as well as co-ordinating the rest of the writing, pictures and demonstration applets. Overall, this collection is made up of around $1'500$ content items – approximately 400 pages of learning materials.

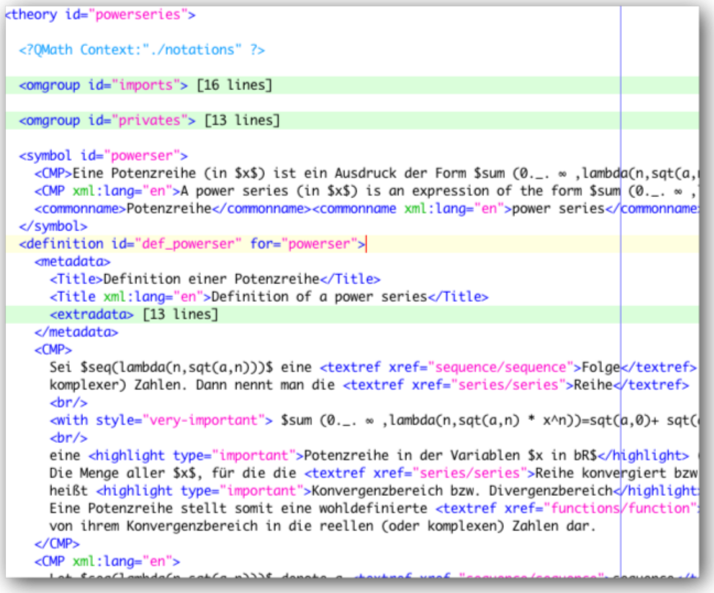
Christian's content sources have been highly customized. Considering the thousands of hours to be spent, the ease of content management was crucial and time was regularly taken to ensure this. The sources were carefully organized by applying measures such as the following:

- Relatively long lines, about a screen width, sliced by hand.
- Topic-based grouping of contents in large files.
- the `import` and `private` elements for each of the `theory` elements were enclosed in `omgroup` elements so that they can be hidden by folding it (see Figure 9.2).
- Large set of own symbols, deemed missing from the existing OpenMath* content-dictionaries, with ad-hoc QMath notations and traditional output graphical notations. These were relevant to domains such as elementary analytical geometry (coordinates, lines, slopes) and calculus basics (sequences and series, convergences).
- Planning of contents around each topic by the maintenance of an Excel sheet with each objective and the details towards its completion.
- CVS repository to allow several people to work together and the other partners of the project to see the achieved content.

LeAM_calculus* is still recognized as one of the most accomplished content collection for ActiveMath. Its breadth has made it the content of choice in several occasions. The authoring experience that Christian gained proved to be successful but several issues are worth noting and have been published in [LG06].

Christian, and his colleague Marianne Moormann, also wrote an amount of interactive exercises with CAS-based evaluation of user-inputs and multiple steps. Both the exercise system and the exercise authoring tools ExaMat* were moving targets during the project.

Christian followed the development branch of the software, being directly involved in the project, and was indeed a very active issue reporter. One dare say that content reload is a fundamental aspect of his authoring activity: some stages of development had the content storage's reload broken and the request was to use the "buildIndex" command to see any new content. This meant for him that it was almost impossible to author, and left him only able to outline a few imprecise plans for content.



```
<theory id="powerseries">

  <?QMath Context:"./notations" ?>

  <omgroup id="imports"> [16 lines]

  <omgroup id="privates"> [13 lines]

  <symbol id="powerser">
    <CMP>Eine Potenzreihe (in $x$) ist ein Ausdruck der Form $sum (0._. ∞ ,lambda(n,sqt(a,
    <CMP xml:lang="en">A power series (in $x$) is an expression of the form $sum (0._. ∞ ,
    <commonname>Potenzreihe</commonname><commonname xml:lang="en">power series</commonname
  </symbol>
  <definition id="def_powerser" for="powerser">
    <metadata>
      <Title>Definition einer Potenzreihe</Title>
      <Title xml:lang="en">Definition of a power series</Title>
      <extradata> [13 lines]
    </metadata>
    <CMP>
      Sei $seq(lambda(n,sqt(a,n)))$ eine <textref xref="sequence/sequence">Folge</textref>
      komplexer) Zahlen. Dann nennt man die <textref xref="series/series">Reihe</textref>
      <br/>
      <with style="very-important"> $sum (0._. ∞ ,lambda(n,sqt(a,n) * x^n))=sqt(a,0)+ sqt(
      <br/>
      eine <highlight type="important">Potenzreihe in der Variablen $x in bR$</highlight>
      Die Menge aller $x$, für die die <textref xref="series/series">Reihe konvergiert bzw
      heißt <highlight type="important">Konvergenzbereich bzw. Divergenzbereich</highlight
      Eine Potenzreihe stellt somit eine wohldefinierte <textref xref="functions/function"
      von ihrem Konvergenzbereich in die reellen (oder komplexen) Zahlen dar.
    </CMP>
    <CMP xml:lang="en">
```

Figure 9.2: The start of a theory of `LeAM_calculus` ready to be read, `imports` and `private` elements folded away.

The metadata* structure has been a domain of lively discussion, the project aiming at a competency-based moderate constructivist pedagogical approach. The many discussions thus lead to agreements on the properties of content items, in the `metadata` and `extradata` elements. This was also followed by discussions on the tutorial component's scenarios. Following the understanding of the metadata*, Christian and his colleagues input metadata for the learning items. But as soon as the tutorial component became a tool that can be used in the software, most of the `metadata` elements were adjusted.

Finally, the fact that many OpenMath* symbols were created is something that one would have liked to avoid or at least something that could have lived hand in hand with the OpenMath community. This would have meant publishing the symbols as OpenMath content-dictionaries (on `http://www.openmath.org/`). However, the OMDoc format being considerably richer, Christian wrote his symbols' descriptions with many mathematical expressions. This made them impossible to export faithfully to OpenMath CDs.

9.3.3 Éva Vásárhelyi

Éva started her ActiveMath authoring in the ActiveMath-EU project, as part of Eötvös Loránd University of Budapest, Institute of Mathematics, Mathematics Teaching and Education Center. She first learned authoring for ActiveMath in the authoring tutorial of January 2007. This tutorial happened to be the one where everything went wrong: all content-storages deployed were reloading constantly the entire content collection because of an incorrect precision of the modification dates of files which happened to have not been tested on the Windows platform. During this tutorial, she could only grasp the broad principles.

Her mission was the translation of LeAM_calculus* to Hungarian which she and her team achieved. Not only did she complete this translation in its entirety, she paved the way for translations of the Math-Bridge project, explaining all the details to be ready before the command *Add a language...** could be applied successfully.

Éva has been a person that suffered from the tacit assumption that an international community has to speak English and long wished for authoring instructions to also be available in German, which would have been normal from the German makers of ActiveMath. However, this was not done until later, when the burden of reading English became a non-issue for her.

In the Math-Bridge project, she and her team encoded the content of the Mathe-Online repository in OQMath using careful and progressive copy-and-paste from a fairly exceptional source encoding: simple HTML with hand-written HTML-encoded mathematical formulæ. As far as we know, this has meant mostly rewriting the expressions by hand or at least considerably polishing them. As of this writing, this means that Éva's team has encoded 300 TeX* pages of textual content with simple MCQ exercises.

9.3.4 Adrien Nicolet

Adrien began ActiveMath authoring in a project called ActiveMath-HESSO in April 2007 at the École d'ingénieurs et d'architectes de Fribourg. His background is that of an apprentice in programming. Adrien discovered authoring by reading the tutorial at `http://eds.activemath.org/` then asking questions, mostly to me directly in French.

Adrien is a typical case of a self-learner with a prior experience in programming and for which the book of tasks brought most of the answers. He found a cross-country way into authoring. Only after questioning and exchanging about the project did he become aware of the difficulty and importance of the quest for semantic mathematical authoring. A bit later Adrien took part in the online authoring tutorial.

In a few weeks, Adrien encoded a book written in MicroSoft Word which is used by self-learners to prepare for the entry-exam at the neighbouring management school: about 30 pages of elementary college mathematics in French with about 20 exercises with optional corrections display. This collection is visible at `http://commons.activemath.org/`.

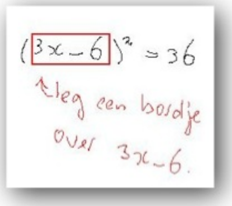
Adrien faced challenges when trying to create content used by the course generation. Similarly to most other authors, only with intensive support did he come through on this. This reached the end of his 6-months training for which he also wrote a Moodle plugin.

9.3.5 Josje Lodder (Open Universiteit van de Niederlande)

Josje joined ActiveMath authoring during the Math-Bridge project where she was part of the Mathematics Department of the Open Universiteit of the Netherlands. During this period, the team of the OU decided to rewrite the content used in remedial courses for the Open University into a more *online oriented* formulation (shorter sentences, more independence of content items...). Most of the content there was to be written in MicroSoft Word, with a small part for use in ActiveMath. Josje is a (practicing) mathematician with a few programming abilities. She has no experience in TeX*, but a bit with HTML*.

She started authoring by participating in a section of the authoring tutorial given at the Math-Bridge's kick-off, May 2009, and working mostly from the book of tasks, several months later. During that writing, she discovered the limitations, or incompatibilities between her view of mathematical documents' rendering and ActiveMath's but only started asking questions in a presentation at a meeting later in January, 2010. Most of the questions in this presentation were legitimate and addressed, but her presentation also showed, again, that the tasks are a sufficient documentation support.

As of this writing, her collection is reaching 10 pages, and contains many formulæ that are semantically encoded and a handful of exercises. Her content items are partially metadata*-annotated; when asked for the reason, she says that she found the templates incomplete and that it didn't seem so important currently.


$(\boxed{3x-6})^2 = 36$
leg een bordje over 3x-6.

Josje's biggest issues in authoring were with the mathematical formulæ. As is typical in bridging courses, aparticular attention is paid to mathematical notations so as not to confuse learners. That care, however, came in to conflict with several of the notation traditions of ActiveMath: Josje expressed wishes to remove the multiplication dot, to have the exact bracket in the right place, and (understandably) to have annotated formulae as in the picture of the right of this paragraph. Such wishes are common among authors but cannot always be honoured because of the exclusively semantic nature of the formulæ: removal of the multiplication dot may make products become words, brackets are generated by the priority mechanism (see Chapter 2), and decorated formulæ such as here have been too hard to safely encode semantically. She has, thus, partially resorted to manual methods (such as the picture aside which could, at best, become a picture realized in a graphics programme) or resigned her wishes.

Her work is ongoing and will be visible on `http://commons.activemath.org`.

9.4 Lessons Learned

Based on the experiences with the authors above, we have learned the following principles.

9.4.1 Importance of the Ease of Update

Historically, one of the first lessons learned was that ease of installation and ease of update are crucial to an efficient support relationship: this allowed myself and other supporters to avoid maintaining multiple versions for us to reproduce the issues reported or to advise on usage.

It should be noted that such updates should not be as untested as the developer trunk. As reported by Barbara and Christian's experience blockages can occur (and did). This is the reason the *stable ActiveMath branch* is maintained and made available to authors.[1]

This ease of update has been used by all authors.

[1]See the documentation of the ActiveMath stable for authors at `http://eds.activemath.org/en/AMinstallation/svn-stable`.

9.4.2 Challenges

Not surprisingly, the areas where authoring has created the biggest amount of questions and discussions is in:

- the mathematical formulæ, stimulated by the requirement of a semantic encoding and by the potential of interoperability,
- the usage of the intelligent features of ActiveMath (particularly in the metadata* for the course-generator and user-model).

For the mathematical formulae, most difficulties lie in knowing what to input for each individual formulae. The *smart paste* approach described in Chapter 7 attempts to help with the discovery of the inputtable content while the WYCIWYG* cycle of Chapter 3 attempts at helping the authors acquire the feeling that an input is correct. The WYCIWYG cycle has been uncountably tested and evaluated by all authors. The smart-paste functions have not yet had a strong testing.

For the intelligent features, difficulties lie in knowing what to input to reach a given effect such as a representative user-model state or a useful generated course.

The approaches to shorten the WYCIWYG* cycle, described in Chapter 3, have attempted to ease this up as much as possible, keeping a focus on the actual usage of the platform as a source of checkable results. As a central contribution of this thesis, lies the conclusion that making this cycle as short as possible is probably the most important area where authoring can be made easier.

The inheritance metadata* was also introduced to simplify the input and management of the metadata. It did not have the time to be widely author-tested.

9.4.3 Precision of Task Descriptions

The first support communications were often oral and typically imprecise with such flaws as a menu-item name being named with a (almost) synonym. Although such imprecision in the language of developers or intensive users of the tools had almost no effect, they had caused great confusion among trainees who discovered language, concepts, and commands all at the same time.

The rule of *same vocabulary* was thus set for the description of the user-interface elements to be manipulated by executors. It seems to be in line with the coherence principle of Mayer and Clark [CM02].

9.4.4 Effectiveness of Visual Documentation

Somewhat extending the lesson learned above, an even more precise description of a task to be achieved is a screen recording: the person recording the sequence of steps typically does not see the significance of certain elements in the same way that the recipients do; keeping an overall view of the computer screen, also including a keyboard view can avoid such discrepancies.

A similar lesson was learned with direct support to authors. For explaining some situations, only screen-sharing was sufficiently explicit.

9.4.5 Importance of Direct Action

As we have depicted in Chapter 3, authors aim at seeing the effects of their authoring actions. This was observed repeatedly with the authors and matches the first lesson-learned described in [AMG$^+$03]: *authoring tools will be more useful if they easily support progressive authoring*.

A related lesson is that authoring actions are only of interest if they are achievable. We have repeatedly met the complete lack of interest in tutorials when we were describing a particular feature with a special condition saying such things as *once we'll have that bug fixed*: authoring actions that do not have an almost immediate effect are considerably less interesting than any other operation that would reach a similar effect without having to wait even if the similarity requires a compromise. These compromises have been numerous and remained too often for an extend period of time. This has sometimes lead to dirty sources.

9.4.6 Central Care for the Source

The approach of text-source advocated in this thesis is often criticized for its over-technical nature. But this approach should really be split in two: the text-editing nature and the principle of a *source document*, being the focus of authoring, from which derivatives are built.

The text-editing nature is common to many systems as described in Chapter 1; jEditOQMath has been intentionally based on a syntax for which a fair amount of support was already available (see Chapter 3). Nonetheless this nature remains criticized by many which parallel the authoring with programming.

The focus on the source documents, as also mentioned in Chapter 1, is widespread and should be stressed as good practice.

The focus on sources has particular implications: the derived documents should be cleanly separated; only the source documents need backup and care. In the case of ActiveMath collections, the source documents included the OQMath files, generally stored in the `oqmath` directory, as well as the source pictures, from which derived images such as PNGs were produced. This separation has been honoured by all authors quoted above and most others. Among others, this implied that all hand-authored pre-recorded books (see Chapter 2) were also stored in the `oqmath` directory.

Eva was one of the people to particularly insist, in her description of *how to start translating*, that the many measures to ensure that the sources are ready before invoking a refactoring such as *Add a language...**.

9.4.7 Learning Authoring Meaning through its Effects

We have observed that authors will only grasp the meaning of concepts of authoring if they can see their effects in the learning environment and can reproduce these. This is in line with the instrumentation process of the instrumental genesis [RB03], whereby users make sense of their instruments as they use them. This is important because it guides what is to be taught to the authors and how the pedagogical competencies of the authors are best measured: in their perceptions and achievements of effects.

An illustration is the following: the reaction of Christian and his team of re-adjusting the metadata* following the availability of the tutorial component's course-generator, even though a *metadata specification* was agreed upon long before and in accordance with their wishes.

Another illustration has been the relatively text-oriented input that Adrien had first used for each of the formulæ, which he then corrected once I explained to him how to see that a formula was delivering its value added services (described in Chapter 2).

Part IV

Outlook

Chapter 10

Conclusions

10.1 Highlights of this Thesis

In this thesis we saw methods and tools to author content that is used for learning opportunities with ActiveMath. The central paradigm coined in this thesis is WYCIWYG* (*What You Check is What You Get*). It proposes a shift of focus in the research about authoring tools: to move away from the editing tools, closer to the intensive use of the learning environment where ways to use the content are explored until an appropriate way to present the content as a learning activity is found.

We have approached a description of the authoring activity by observation and have described the rich set of features that ActiveMath, our target learning environment, offers.

The central authoring practice described in this thesis exploits three sorts of tools: the widespread file management tools (and their extensions as far as versioning systems), the editing tool jEditOQMath, a popular text editor, and the ActiveMath learning environment itself. The tools are enhanced to communicate to each other, making together a tool-set that allows the WYCIWYG* paradigm and its exploitation for a community of authors that can re-use content and help each other.

The research investigated in the thesis is a beginning in many respects. Nonetheless, it has allowed large amounts of content to be created. In the remaining part of this conclusion, we sketch future research avenues that will extend the research of this thesis.

10.2 Open Research Questions

In this section, we review the research avenues opened by this thesis; these are future works that are beyond the scope of a single chapter. We attempt to note whenever relevant work exists in this direction.

10.2.1 Generalization of the Re-use Actions

The re-use model we have described in Chapter 6 focusses on a precise sense of re-use: that a physical copy of the content collection is made and included. Many other interpretations are sketched in this chapter and we stated why the re-use relationship appears to be useful for others to evaluate the changed or used content: it provides another perspective on the applicability of the content and may thus invite others to re-use similarly.

Based on our experience in the i2geo* project, reported in [Lib11], it appears that the forms of re-use described in this chapter are limited in scope. For example, they do not include all the cases when a physical re-use was attempted but then abandoned in favour of a remade version. Such scenarios as the *imitation of content* are described in [Lib11] as being common among teachers for which the usage of a computer based tool is only valuable if it does not carry too big a side cost. For example, if re-using a collection is creating a large work at filtering for the benefit of only a few added items, the concrete re-use will be abandonned in favour of a self-grown *shallow* copy, but the relationship of re-use would remain.

Shallow copies, and probably more generally the notion of *being inspired by some other content* remain forms of re-use which are also desirable to be displayed. This generalized relationship should be investigated in the workflow of authors that use the content in concrete learning opportunities. An ontology of such relationships, probably extending the *provance information*,[1] how they can be input or automatically detected and how they can influence the workflow of using learning resources, should be investigated. The special nature of ActiveMath content does not make it an exception; rather, it makes it a content format that is somewhat more likely to be re-used.

10.2.2 Visual Authoring

The most common critique to the softwares of this thesis is the lack of an authoring tool that is visual. While we have clearly shown in Chapter 3 that a term often coined as desired, WYSIWYG*, does not make sense, a form of tool that would be less markup oriented and more visual is desired and we cannot dispute this.

[1]The Provenance Data Model is an effort to annotate using semantic web techniques, multiple versions across the web, see the working group web-page `http://www.w3.org/2011/prov/`.

In fact, several implementations have attempted to address this mission but none have concluded. The major stumbling blocks of these implementations is that they reached a fragile capability of displaying but needed a full revision as soon as it was made clear that the feature of multiple undo operations was needed. Three attempts of visual editors were realized by different persons under my direction. All were discontinued for various reasons – the biggest of which may have been the lack of funding or clarity in the mission.[2] The authoring tool ExaMat*, as described in Section 3.3.5 appeared as the most promising but never reached generalization to all item types or to whole collections.

The set of features for such an authoring tool to be useful in a similar way as jEdit-OQMath is not small. We have outlined in Section 3.9.1: it includes short reload cycles, the ability to travel back, to report errors, to paste from external sources and to include the objects of the authoring tools in online communications.

Of course, it should be an authoring tool that is complete in its ability to edit content for ActiveMath or even to edit any OMDoc files. This completeness and the moving target constituted by the OMDoc encoding and the metadata of ActiveMath has prevented a revival of any existing approach to visual editors.

Nonetheless, an authoring tool that would be tightly coupled to a content project, with well identified users that would eagerly support the creation of the use cases of the tools, with a carefully identified knowledge structure, plus a fixed ActiveMath server version has a chance to be realized. It should be tightly integrated in the learning environment (enabling direct publication and preview) and in the re-use and publication workflow (hence it should support a collaborative versioning system). It could be based on such technologies as semantically-annotated HTML*-editors, which would enable the communicability and the visual nature; it could exploit such a paradigm as that of Plato [AFNW07]: a gradual semantic enrichment of presentation where the semantic is only visible as annotations.

10.2.3 Rich and Evolving Knowledge Representation Structures

One of the central assumptions that has lead the development of software in this thesis is the flexibility of the knowledge representation: from the start onwards, extensibility of the mathematical symbol set was a requirement; the perspective of OMDoc as an extensible language (inheriting from the XML* perspective) was permanently visible; along the projects of ActiveMath, multiple versions of the metadata specification were edited, each based on a particular view of the pedagogical values of learning objects. At least the projects In2Math, Matheführerschein, LeActiveMath, and Math-Bridge had different metadata schemas, each in several versions, many with implementations depending on them. This has meant that many documentation efforts or software efforts became quickly outdated. Our

[2]These three were called `OmdocJdomAuthoring`, `ProtegeAMMetadata`, and `Quoniam`. Two of them are in the `AMauthoring` repository: `http://svn.activemath.org/AMauthoring/trunk/projects/`.

group has simply considered an update of the *core* documentation: the metadata specification, a simple dictionary of names and possible values, and the DTD*.

A few research attempts address the limited capabilities of the DTD* by producing other markup schemas, but always under the perspective of a validation of the authored content. Indeed, an XML or Relax NG schema* does offer a slightly finer grained grammatical control, for example, the ability to declare that it makes sense for a theorem to also be a definition.

The goal of validating the sources should be expanded to multiple **other exploitations of one central declaration of the knowledge**:

- The dictionary approach of the *metadata report* should be included as a documentation produced by the declaration of the knowledge. Knowledge structures such as OWL* ontologies or XML schema* make it possible to generate a set of HTML* pages documenting each of its elements (`owldoc` and `schemadoc` are two example tools). The HTML* pages can then become also useful hypertextual references which can be easily shown to others and be linked from example content fragments presented in documentations.
- The declaration should allow as much as possible inference. For example, if it would be possible to encode the educational level (the *educational context* using the language of LOM* and the ActiveMath metadata) in an ontology, it would be possible to infer country and age, and to generalize to similar educational levels of neighbouring regions. Following the example of the metadata inheritance in Chapter 8, we recommend that such inference be experimentable with such an approach as a preview that shows, in the authoring context, the results of the inferences as a readable authoring representation.
- It should be possible for the ActiveMath learning environment to exploit the knowledge representation as a source of possible values for some of the properties. For example, the search tool does not hard-code the set of values of the *difficulty* metadata field but reads it from the DTD*. However, we acknowledge that the knowledge representation does have implications on the software and it would be illusory to expect any change of the representation to be compatible with ActiveMath. For example the removal or renaming of a property from the knowledge representation may well make the course generator scenarios and the search tool's user interface unable to run.
- The validation process should generate errors with hints that are helpful and understandable by the authors; this is particularly visible in schema validations that try to be pedantic about such constraints as the set of children, in particular XML-schema* which shows element-names with namespaces.

As we have sketched above, candidates for the implementation of such a knowledge structure include OWL* and XML schema*. Both are mature technologies in wide use in other contexts. OWL has the possibility of doing considerably more inference and merging multiple data formats while XML Schema has more robust implementations.

A final requirement to the proposed research is the support for knowledge structure version changes. It has become clear along the years that a good authored content collection is here to stay across multiple versions of the software and across multiple versions of the knowledge structure. This, essentially, means that the content needs to be updated everytime the knowledge structure changes. The current approach involves upgrading the DTD* files then validating each of the sources and repairing where necessary. Such repairs could partially be made automatic or semi-automatic (for example, proposing choices between candidate updates) and documented. It is not clear yet if it is recommendable to offer transparent upgrades for older knowledge structures: the differences in software behaviour may be surprising. However, a recommendation to go to the authoring tool and make sure the changes are acceptable may be safer.

A major advisory group of the World Wide Web consortium has left an amount of open-issues and published an amount of best practice about language evolution and authoring `http://www.w3.org/mid/4DA4A09D.6090806@arcanedomain.com`.

10.2.4 Mathematically Fuzzy Formulæ Search

As we have described in Chapter 5, a search option for formulæ appears to be desirable for many learning objectives, but it is not very clear how to offer **tolerance** (a core feature of a search engine) in the search of mathematical formulae.

In parallel to our research on mathematical search engines, a few efforts have been made in this direction. [You05] considers the TeX*-tokens to be elements of the flow constituting a formula so that x^2 matches $x + y^2$; we believe this would be highly confusing for a learner. [SL11] considers sub-formulæ as elements of a formula allowing, at worst, arbitrary sub-terms of the query to match (e.g. $\sqrt{b^2 - 4ac}$ to match a document with $4ac$ only). None of these approaches are mathematical approaches to fuzziness; the only one is the canonicalization attempts reported about in [AY07] and the usage of canonicalizations such as in [MM07].

For example, it would be normal, in most cases, to consider $a+(b+c)$ to match $(a+b)+c$ but neither a TeX*, nor MathML*-content, nor OpenMath* token-sequence is common. It should be possible to exploit the large set of tools and theories about rewriting systems to enrich the search index with alternatives of any given mathematical expression in a way that is adequate for the learners being considered. Such alternatives should clearly be less preferred than the exact matches. Examples include:

- In the context of users that are mature in the domain of elementary algebra, all of its rules should be tolerable (commutativity, associativity, ...).

- Users that are familiar with trigonometry are likely to be satisfied with a match of $\sin(-x)$ for $-\sin(x)$ but learners that are in the midst of learning it would certainly not be.
- As indicated in [MM07], expansions such as to find documents containing $\sin^2 x + \cos^2 x$ when searching for 1 are likely to be surprising to all although some very specific tasks may make use of it.

Since we are talking of finding formulæ in the documents that are only mathematically equivalent to the user's query, it is imperative that the formula that is found to be equivalent is highlighted. This feature has never been implemented in the search tool of ActiveMath (or almost any other mathematical search engines). Simple highlighting might even be insufficient for a user to see the relationship of a match between $\arctan x$ and $\int \frac{1}{x^2+1}\, dx$.

Generating such alternative formulæ would be based on replacement rules that are mostly pulled from the Formal Mathematical Properties (`FMP`) elements of the OpenMath* content dictionaries. Designing the generation in this way may be a good way to ensure extensibility of the set of symbols and ongoing compatibility of the search tool. It should be noted, however, that choosing which equivalence is desirable is often more a pedagogical question than a purely mathematical one so that the selection of the applicable rules should be conducted with a specific learning opportunity in mind.

Moreover, as we have sketched above, it is likely that the past actions of the learners with the learning environment are highly relevant to the decision of applying or not a given substitution. A probabilistic approach ressembling the user model should be used to decide such. The user-model's competency-approach described in Section 2.4.4 runs a risk to be inapplicable. In particular, the propagation among evidences is likely to be inappropriate to model the mere awareness of a substitution rule.

The approach taken by the FuzzyQuery class of the Lucene* library explained in [HG04] is probably relevant here: it computes the rewrites up to a "fuzziness distance" at query time (the Levenstein edit distance), based on the tokens found in the index then queries for the disjunction of all these terms, weighted by the distance. This allows exact matches to be preferred and *far-away* rewrites to match but come later: in a corpus without exact matches they would appear on top, as a useful disambiguation service but they would only partially hide the exact matches if available.

10.2.5 Mapped and Compiled Remote OMDoc Storage

The description of the content storage of Chapter 4 has introduced a high-performance engine based on an intensive indexing-process done at load time. This indexing process has been, however, recognized as imposing long wait times in periods where authoring could happen (see, e.g., Section 9.3.2).

Long indexing times typically appear when one adds a new collection or when one performs massive changes in a collection and, more often than it should, when unpredictable storage issues emerge. As described in Chapter 6, adding a new collection, or switching on and off its usage, could be a very regular activity when attempting to discover the right content collection to be re-used.

The storage chapter describes several phases before the indexing and decomposing these steps may provide a way to avoid the lengthy waiting time. One of the heavy steps is the simple storage in index, another is the reference resolutions (the *absolutization phase* as well as the computation of the reverse relationships).

A first step to enhance the storage speed could be to distribute, with the downloaded collections, the binary data of their index. However, this runs a strong risk of versioning incompatibility when transporting between ActiveMath servers of different versions.

An alternative could be to store *compiled* OMDoc files with the collection where all references are absolutized*. It is not clear whether that would break some of the possibilities of authors, for example, to allow loaded collections to rely on a revised set of symbols as explained in Section 4.7.3 but it would surely make the indexing process lighter.

A combination of these two could be technically possible in a web delivery environment: files that are compiled could be enriched with XML-maps that indicate the byte-position of the start and end elements of each elements of interest and their ancestors. To respond to a MBaseRef query such as `getTextualContent`, an XML* parser would only need to parse the given content fragment, something which is currently being done in a sufficiently speedy manner from the index's storage at each query. The indexing of relationships would be stored within the compiled XML documents and would be parsed similarly.

This mapped extraction process is particularly interesting because it can be executed over the HTTP* protocol: indeed, this protocol defines a header called `Range` which enables one to specify which ranges should be delivered in the response. Most file-based HTTP servers we have met (Apache, IIS, Tomcat, NGinx) implement this header. An author could, thus, publish his compiled OMDoc files within a simple file serving facility, as is commonly available and cheap to purchase and authors that want to to re-use this content collection would only need to give to their authoring ActiveMath the URL of that compiled collection. The map information would be downloaded fully (this is probably 1% of the content collection) and only the content items that are actually used by the ActiveMath learning environment would be downloaded and cached by the mapped-XML client.

This would support the simple *shopping* activities of authors who would be able to try within their ActiveMath the re-use of some content collections without actually downloading it in full. A full implementation of this idea would enable the load of one content collection found on the web in less than a minute, something which is far from being realistic currently. The lightness of such a shopping activity would enable the trial to load multiple collections, and multiple versions of them, before settling on one and downloading it in full.

Such an advanced feature comes with several challenges. One of them is the predictability of such an easy loading compared to a *traditional* loading: it is likely that, for example, some reference resolution would differ. Another challenge resides in measuring the impact of a loading:

- Does it redefine some notations? If yes, one needs to recompute the presentation pipeline's XSLT* and invalidate the rendering cache of affected content items.
- Does it redefine some typical course generation? If yes, the author should be aware of that and revise the result.
- Does it redefine typical search results? Similarly, awareness should be provided.

Such a mechanism may have refinements allowing simple revisions of the downloaded content. One could enrich the loading with a filter to only accept some items (for example only the notations, or only the exercises); one could also enrich the loading by inviting an author to revise a document that he loads locally.

10.2.6 Authorable User-Model and Tutorial Components

We have described the WYCIWYG* paradigm in Section 3.6 which relates the editing work to the verification task as a central cycle of the authoring activity. The challenges that authors faced to make good use of the learner model and the tutorial component, as described in the Chapter 9 remain. An attempt has been made by proposing the metadata inheritance in Chapter 8 but it is likely to remain a challenge until a WYCIWYG cycle is possible with these two components. One could imagine the provision of the following features:

- a way to re-start with the learner model and book list at a fixed state
- a one step way to trigger a course generation with the same objectives and the newly loaded content
- a way to store and communicate such setups so that others can reproduce the cycle

Such features should be weighted by the pedagogical value of the scenarios they make possible to author. They are focussed on the course generation scenarios but many others that use these components make sense: for example, a different set of features would support the development of content that could be used with such didactical missions as *succeed in as many exercises as needed to get the mastery bullets (Section 2.4.4) all green*. This is the activity that was running when the top photograph on the cover was taken.

Part V

Appendix

Appendix A

Software Contributions

In this appendix, I detail the software contribution of this thesis. I point to the places to find software and describe the contributions of the developers.

A.1 Overall Architecture

The authoring experience described in this thesis, mostly in Chapter 3, is based, on the one hand, on editing (jEditOQMath) and management tools (file management, versioning, e.t.c., which we do not describe since these tools were used in a straightforward manner) and, on the other hand, on the learning environments which can *exploit* the content as it gets repeatedly refined."

A.2 jEditOQMath

jEditOQMath is an assembly of multiple softwares which I have re-used, which I have written, or the implementation of which I have coordinated. It is delivered as an application started with one click and is described in Chapter 3. jEditOQMath is made of the following components:

- **jEdit** is a widespread Java-Based text editor written originally by Slava Pestov and now maintained by a group of other contributors. See `http://www.jedit.org`.

  My contribution to the jEdit core, except for a few debuggings, has been to provide the abbreviations inspired by the MathML entities as explained in Section 3.4.2. My other software contribution, which did not get reused in the jEdit project because it required endorsement of plugins made by others,

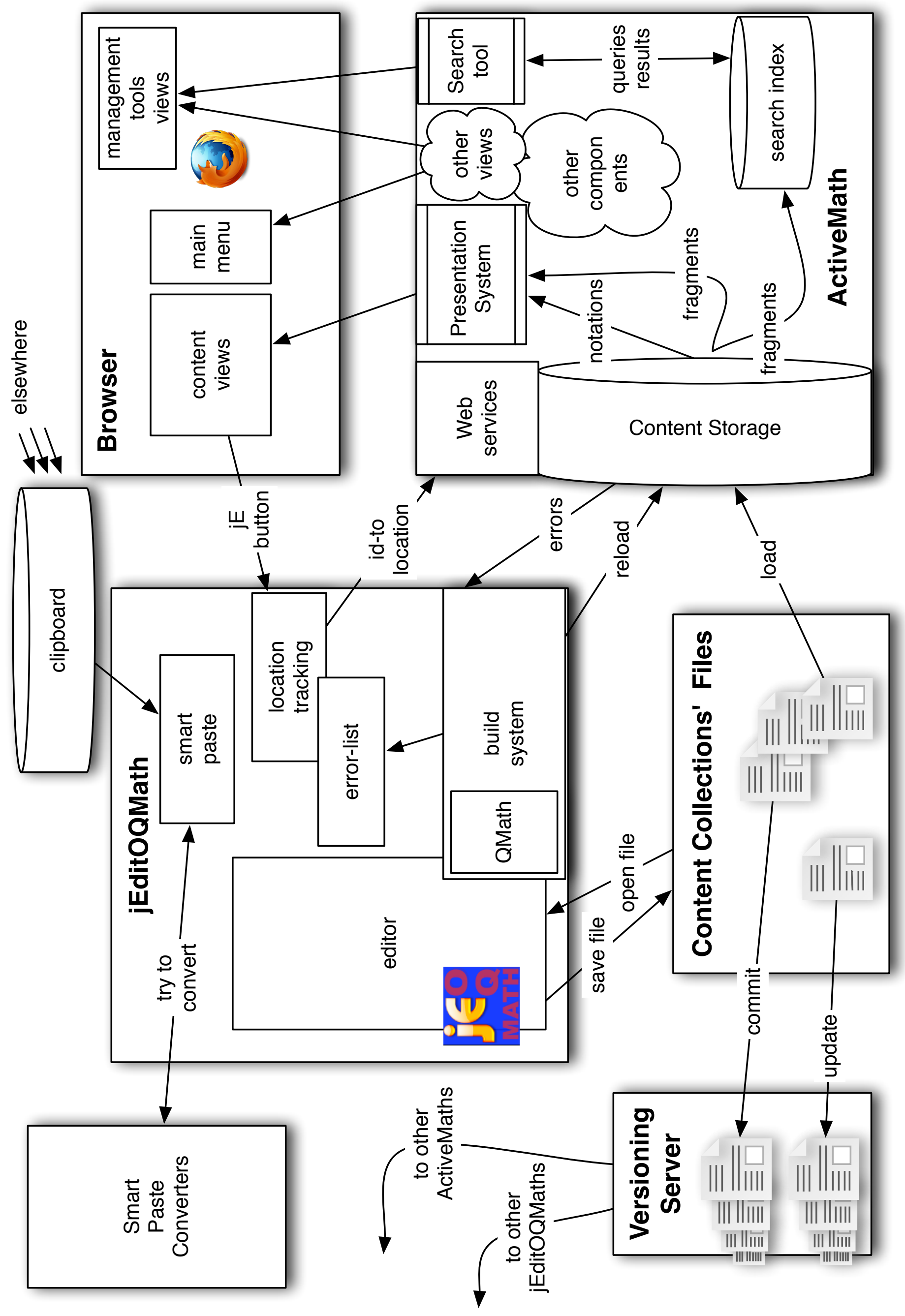


Figure A.1: The overall architecture of the authoring setting of this thesis.

is that of a packaging that can be self-updated: a build-system that encompasses the very many components described below (based on Apache Commons Jelly scripts), and a delivery as a JNLP* application (also named *Java Web Start*) so that a user with a *regular* desktop is able to start and update his jEditOQMath installation in little time.
jEditOQMath is built by checking out its project at `http://svn.activemath.org/AMauthoring/trunk/projects/jEditOQMath/` and invoking the Apache Maven 1.1 build (`http://maven.apache.org/maven-1.x/`).

- **jEditOQMath** is delivered with a selected list of jEdit plugins mostly also realized by Slava Pestov: AntFarm, Archive, CharacterMap, Console, ErrorList, InfoViewer, JDiffPlugin, JellyJEdit, SideKick, SwitchBuffer, Templates, XML, XSLT, and XMLIndenter plugins. Most of jEdit is under GPL*, the jEdit adjustments included; the packaging procedures are under MPL*.

- **OQMath** is a Java programme that wraps the QMath native executable of Alberto Gonzáles Palomo. This java programme is available under MPL* from `http://svn.activemath.org/AMauthoring/trunk/projects/OQMath/`.

- the **OQMath Plugin** is a plugin to concentrate the extensions of jEditOQMath in one menu. Being built on the jEdit interface, it is under GPL*

- **omdoc2oqmath** is a java library originally written by George Goguadze then maintained by myself and Yecheng Gu in order to parse QMath context files and to convert OpenMath formulæ to their (shortest) QMath form with the given notation definitions.

- **Smart Paste**: a thin library that responds the *smart paste* command inserted in jEditOQMath written by Yecheng Gu under supervision of Eric Andrès and myself. The smart-paste function uses the smart-paste converter service which integrates several conversion tools including the Blahtex processor (`http://www.blahtex.org/`), the WebEQ developer tools (`http://www.webeq.com`), and the Wiris Input Editor ([MEC$^+$06]).

- **Searchable Item List**: this component is a small utility intended to be integrated in jEditOQMath and that can live by its own. It is written by me and is available at `http://svn.activemath.org/AMauthoring/trunk/projects/IdSearch/` and is available under MPL*. See Section 7.6.

- jEditOQMath also comes predefined with **templates** meant to help the starter. They are written in Velocity* and are part of jEditOQMath source. They are under MPL*.

- A central piece of the WYCIWYG workflow, the **build-scripts** are run by Apache Ant (`http://ant.apache.org/`). They contain classical tasks (such as copies), and call to dedicated tasks part of jEditOQMath: the StaticBookMaker, OQMath, and the MBaseReload tasks. The Ant scripts are constructed at start-a-collection time by a script run in Apache Jelly (`http://commons.apache.org/jelly`. All of them are under the MPL*.

- The **OQMath jEdit plugin** embeds the functionalities above (smart paste, start a collection, the notations-list, ...) and includes a few extra commands such as the possibility to see the "OMDoc compilation result" matching the current place in OQMath, or the refactorings. The plugin is under the GPL*.

A.3 In ActiveMath

Another part of the software contributions of this thesis is embedded in the ActiveMath server. We describe the various parts below.

Being part of ActiveMath, all of these software contributions are under the DFSL*. Some parts, under the `org.activemath.omdocjdom` package, are under the MPL* as well.

They are all available in the ActiveMath source tree, see `http://www.activemath.org/Software` or `http://eds.activemath.org` for availability.

A.3.1 Content Processing Model: OMDocJDOM

In order to be able to manipulate the OMDoc XML* documents, and to implement their behaviours, I have written a library in the package `org.activemath.omdocjdom`. That library extends the JDOM* library for in-memory representation of XML documents which is well known to offer one of the easiest application programming interfaces to XML documents (see `http://www.jdom.org/`). OMDocJDOM extends it with:

- commodity access and creation methods (such as `getMetadata`)
- identifier relative resolution following the `mbase://` URI-scheme (described in Chapter 4)
- dedicated subclass objects for each element type
- SAX*-based parser utilities
- key-value-pairs representations for the metadata* of ActiveMath items
- access to content-collection information given by the content-descriptor
- support for line-number-tracking

A.3.2 Content Storages

The content-storage of ActiveMath is described in Chapter 4, it offers an abstract access to the OMDoc content specified by the `MBaseRef` interface (`http://www.activemath.org/~ilo/redirs/MBaseRef.html`) that I wrote with Martin Fuchs.

Several implementations of the `MBaseRef` interface are available:

- the current content-storage interface is called **SLuMB** (serial Lucene MBase) and is based on an index built using Lucene*. It is in the package `org.activemath.omdocjdom.slumb` and `org.activemath.omdocjdom.slumb.data`. It takes as input a set of content-descriptors and their associated collections.
- multiple **historical** implementations are left since the implementation of the content storage has been a critical part of ActiveMath's performance. The first two are not in the source tree anymore – the org.activemath.omdocjodm.mbase package provided the first validating implementation and the last full-in-memory implementation, org.activemath.omdocjodm.thembase attempted to provide connectivity to the Mozart-Oz-powered MBase [FK00] (with inacceptable performance), and org.activemath.omdocjodm.exist attempted to provided connectivity to the eXists XML database (`http://exist.sourceforge.net/`, failed due of the complexity of OMDoc documents).
- a **cached MBaseRef** in org.activemath.omdocjodm.cache wraps another MBaseRef implementation and stores the last 1000 results for faster retrieval the next time.
- classes for **remote MBaseRef services** through XML-RPC: `OJXmlRpcMBaseRef` and `SluMBaseXmlRpcServer`

All these implementations are written by me except LuceneMBase, the eXist attempt, and the CachedMBase, which are all written by Shahid Manzoor.

A.3.3 The ActiveMath Search Tool

The search tool is described in Chapter 5. It is made of an index built in Lucene* and uses dedicated analyzers for OMDoc. It is contained in the packages:

- for the back-end: `org.activemath.omdocjdom.index` and sub-packages `analysis` and `queries`. These packages are written by me with touches by Dominik Jednoralski to accomodate his work on latent semantic indexing in sub-package `infomap`
- for the user-interface, a set of Velocity* files have been written by Dominik Jednoralski and myself in directory `webapps/ActiveMath/search`
- for the user-interface control and data-objects, the package `org.activemath.-webapp.dict` and `queries` sub-package that I wrote.

A.3.4 The Notations

The tools to process `symbolpresentation` and `notation` elements of the ActiveMath encoding of OMDoc documents are in the package `org.activemath.presentation.xsl.symbolpresentation`. It has been written by Shahid Manzoor and is maintained by myself.

The set of notations for mathematical formulæ is in `omdoc1/cd` and has been written by multiple contributors of the ActiveMath group, including Shahid Manzoor, Michael Dietrich, Christian Gross, Abdelshafi Bekhit, and myself.

A.3.5 Copy and Paste by Drag-and-Drop

The application of the copy-and-paste paradigm is implemented in the learner's browser by way of a drag-and-drop action within the ActiveMath presentation architecture:

- in the presentation-pipeline's pre-processors and in the XSLT files (see Chapter 2),
- as the *clip-controller* to serve the objects at the given URL in the package `clip`
- and as javascript to highlight the terms and propose the popup menu.

The original implementation was by Vladimir Brejnev, in the in2math project, the current implementation was done by myself and, for the javascript part, Dominik Jednoralski.

The other part of the implementation is the reception of the dropped links which is a feature of the Wiris Input Editor (see Section 2.4.6) implemented by Dani Marquès in collaboration with myself.

Appendix B

Software Functions

We separate the functions in the three spaces that we described in Chapter 3: the files, the learning environment, and the editing.

B.1 Files Management Functions

Authors are creating their work within files which they can manipulate in their file-management environment (the MacOS Finder, the Windows Explorer, GNOME's Nautilus, or simple Unix shells, ...):

- they can view files by name and rename them
- they can organize them by folders
- they can open these files by double-clicking on them which generally launches the appropriate editor (the installation process of jEditOQMath associates itself with this action for `.oqmath`, `.omdoc`, `.properties`, and `.xml` files but this association, as service of the Java Web Start client, has often been buggy)
- they can open server files to run them: this is how ActiveMath can be started and stopped

Each of these environments can be enriched with an ability to synchronize each folder to one in a subversion repository. Extensions for theses include TortoiseSVN and SCM-plugin. The file-management display is enriched with witnesses of the up-to-date, to-be-updated, or not-committed statuses of the files.

B.2 Authoring Functions in ActiveMath

As part of the recommended set-up for authoring (see Section 3.2), each author is expected to have his own learning environment where he can load content collections and see them at work.

Thus far, such *authoring ActiveMath* has been installed on each author's authoring machine.

B.2.1 Use as the Learner

Among the crucial aspects of authoring is the verification that the content within the learning environment offers a valid learning opportunity: to this effect, authors use the ActiveMath learning environment and perform each step a learner would perform while learning; they then verify that the information presented is sufficient and appropriate.

Though there is no specific requirement of this usage, a regular ActiveMath server is needed with a regular user account – this function is fundamental and is likely to be used with a user account that is different than an authoring user account.

Beyond the use for verification, authors are likely to become power users of ActiveMath, especially when it comes to evaluating content made by others. Thus, they use the search tool of ActiveMath and most of the details it provides about each content item. See Section 5.1.2.

B.2.2 Functions for Author Users

As part of the normal installation process of ActiveMath, a user has been registered with authoring privileges so as to be able to manipulate content. This privilege adds the following features in to the user-interface of ActiveMath. Most of them are available through the tools menu.

jE button this link is displayed aside of each content item. It is a button that allows, in classical conditions, to request jEditOQMath to open the source of the content item: jEditOQMath, on receiving this command, requests from the content storage of the authoring ActiveMath to know in what file and at which line the content item is. If an OMDoc, an attempt to find the corresponding place (the same line number) in the OQMath source is made. See Section 3.4.4 about this button.

The Symbols Presentation Tool This tool is to inspect the available notations, and to explore the rendering of an individual expression. Users can select to list all the notations of individual collection, in each format and each language as well as input an OpenMath expression and observe its rendering.

A drag-and-drop from a notation that is listed can take the user to the notation source. Such an inspection is particularly useful in understanding the conflicts of notations which can occur easily because notation templates are made of an OpenMath expression that can be complex (see the section about the presentation pipeline, Section 2.4.7).

buildIndex A command-line tool to trigger the build of the content storage index. This command is equivalent to shutting down the server, deleting the `data/slumbdb` directory, then starting the server again.

MBaseRef Tools Although the classical edit-save-reload-inspect cycle described in Chapter 3 is meant to reload the content storage as triggered by the reload function of jEditOQMath, there are times when inspecting the content storage remains useful, in particular when working on servers that are not authoring servers (e.g. for school servers or group-shared servers). This is where the MBaseRef tools come into play, a tool that is available for authors and administrators of ActiveMath.

It allows the user to list collections, theories, and items, and see the source that is stored. It also allows the user to request the reload of a collection, reporting errors in the browser.

Within an ActiveMath server meant to be shared in a group, the MBaseRef tools' reload function can be enriched by a *hook*: this hook can be the update operation of a checkout. This allows an author to *display* the result of the content he just committed (published in the versioning server) to the shared server. This function is activated on `http://commons.activemath.org/`.

B.3 Editor Functions

We list here the functions of the editor that are supplementary to the traditional editing commands and relevant to the authoring practice.

The jEditOQMath distribution, aside of having the specialty of being distributed over Java Web Start, includes an almost standard jEdit, Version 4.2, accompanied by assorted plugins that are relevant to our activity: Ant Farm, Archive, Character Map, Console, ErrorList, InfoViewer, JDiff, Jelly, MacOS, OpenMath Presentation Editor, SideKick, SwitchBuffer, Templates, XML, XMLIndenter, and XSLT. Almost each is used in particular commands below. The central plugin to the extended functions is called the OQMath module and is displayed in a separate menu.

B.3.1 Navigations Functions

Search ID... (and Search ID on word): A command of the OQMath plugin that displays a small window to help insert the right reference: this window gives the focus to a text-field which allows us to search content items and go to them or insert reference to them.

See Section 3.4.2 and a picture in Figure 3.3.

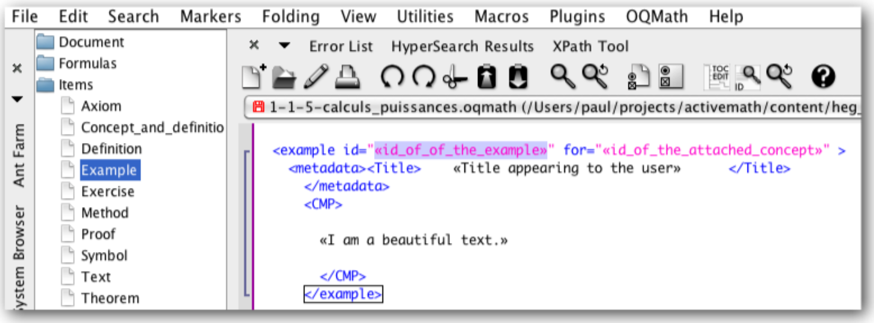
Goto matching place in OMDoc dest and **Goto matching place in OQMath source**: are two functions to switch the buffer of jEditOQMath between the OQ-Math source and the OMDoc source, keeping the same line number. This brings the user to the place in the destination that was output from the same place in the source since the OQMath process keeps lines intact as explained in Section 3.5.2.

Drag-and-drop of References : Hyperlinks to ActiveMath content items (e.g. in most titles rendered in web-browsers) can be dragged from the browser and dropped on to jEditOQMath: if on the editing-pane, a reference is inserted, if outside the editing pane, the content item is opened.

B.3.2 Input Helpers

Start a collection... : This wizard presents a dialogue asking for the name of a content collection (a string without spaces or slashes) then populates that content collection with template content:

- A directory with the collection name is created in the authoring ActiveMath's content folder. It is populated with the relevant files and directories
- A content-descriptor file which declares a default static book and a default grouping for the course generation (see Section 2.3)made of all the content items found in the collection. This is complemented by an Ant build-file so that this collection can be processed and reloaded from ActiveMath.
- A directory `oqmath` with a first OQMath source, called `first.oqmath` and containing a single *hello-world* content item, a directory containing the current default set of DTD* s, an empty directory `pics` and an empty directory `omdoc`.
- Finally, the creation process copies the content-descriptor in the `conf` directory of the authoring ActiveMath (thus enabling this collection) and runs the `build-offline` Ant-target of the *Reload** process which creates the output: the first OMDoc file. The author is then advised to restart his ActiveMath which allows the newly built content collection to be immediately available.



```
<example id="«id_of_of_the_example»" for="«id_of_the_attached_concept»" >
  <metadata><Title>    «Title appearing to the user»      </Title>
    </metadata>
    <CMP>

      «I am a beautiful text.»

      </CMP>
    </example>
```

Figure B.1: An example template and a part of the templates list on the left.

The collections templates are configurable. Their template files (using the Jelly plugin and the Apache Commons Jelly template engine) are stored in the settings directory of the user-home so that the author can modify it to suit his practices. This command is also best used to create a collection that has an updated DTD* and content-descriptor corresponding to the new standards.

Templates : One of the tabs on the left of jEditOQMath is called *Templates*. It inserts a new content item at the cursor, or other typical fragments. Each template is followed by positioning the cursor at the beginning of the inserted template and activating the following command *Select next template zone** so that the first normally relevant place to modify is selected and the user can modify it. Typically, for content items' templates, the first selected template-zone is the identifier.

A part of the templates list is in Figure B.1.

Select next template zone : This simple search function selects the next fragment of text surrounded by french quotes. This works well with templates that build an almost valid fragment (e.g. which can display) with islands surrounded by french quotes in each place where an input is expected. The template zones contain text that describe the expected input while this command, Select next template zone, selects it fully so that the author just starts typing after glancing at it to provide the necessary input.

An inserted template, for the item type example, is shown in Figure B.1.

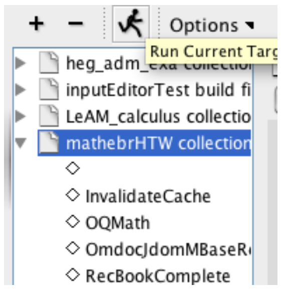
QMath Experimenter... : a tool to try the QMath conversion by inputting an expression and getting QMath. The conversion is operated with notations of the currently edited OQMath document. The QMath experimenter, after the author has pressed return, displays the resulting OpenMath, in a formatted way, so that an author can verify that the notations are correct and the QMath processing of them is as expected.

Notations list... : A tool to parse the notation context directives, and the directives they include, summarizing the result as a sortable table of notations. See Figure 3.4.

Smart Paste... : A specially enriched paste function that attempts to convert the content of the clipboard by looking at it and invoking online services. See Section 7.6.

Validate XML... : this function invokes the classical validation of the currently loaded document expected to be an XML* document. OQMath documents are XML documents and this function is operated every time the user saves. This allows authors to be on the safe side, making sure that the basic structuring rules of the documents are right or they are displayed with an error as reported in Figure 3.6. This validation is, however, not replacing the check of the result in the learning environment.

B.3.3 Suggestions, Preview, and Reload Functions



Reload : As a central role in the WYCIWYG cycle, the reload function is invoked by the author by clicking on *the running man* icon, having selected the build-file of the right collection.

The reload process performs compilation and scan operations on the changed files of the collections and reloads them from ActiveMath; each can trigger errors. The process is described in Section 3.4.3.

Generate Imports An interface to the content storage to display, for the `theory` element that contains the current cursor, suggestions of the `imports` element which would resolve the references of the current document. See Section 4.7.3. In effect, the generate imports command performs a dry-run of the reference resolution mechanism for the given item, outputting the references that need resolution and searching how to resolve them in the existing set of items.

Preview Metadata Inheritance : An interface to the content storage to display the result of the metadata inheritance process documented in AM: 8.3.

B.3.4 External Tools

Assembly Tool : a menu item to launch the assembly tool to edit local files using the visual editor described in Section 2.5.2.

Wiris Domain Editor : a menu item to launch the domain editor to edit the domains available on the local ActiveMath. More information about the Wiris input editor and its domain editor can be read in Section 2.4.6.

B.3.5 Refactoring Functions

Clean-up OMDoc A method to remove *extra* attributes from the XML* tree: those attributes that are in the DTD* with a default value that is the same. These attributes, which are needed by XML processors that do not use the DTD* have been recognized as the main source of noise in OMDoc sources that enter automatic processing.

This command and the next are useful when bringing into an ActiveMath collection such documents as those produced by the ExaMat authoring tool (see Section 3.3.5).

OMDoc 2 OQMath A processto transform an OMDoc source back in to OQMath: it converts each OpenMath expression to a QMath expression (using the same conversion method as in the *Smart Paste*...* function) then processes and cleans up the OMDoc so as to get a document that is close, in readability, to an OQMath source and that processes correctly in the current setup.

Apply XSL... A feature to allow technically skilled helpers to support refactoring some of the OQMath sources by providing an XSLT* stylesheet that authors can apply on their sources. It can be used for operations such as "upgrading the schema" where typical values have to be adjusted: such adjustments are often specific to a content-collection and thus cannot be part of the server. Instead it is part of a service to the author of the collection, which verifies the result.

Add a language... Invoked by authors starting the translation of an OQMath file into another language. This command duplicates each element that has a language attribute (e.g. `CMP`, `Title`) of the source language within the OQMath source. The content of the element is not translated but prefixed with an order in the given language (TRANSLATE-ME, FORDÍTSA-LE-NEKEM, MICH-ÜBERSETZEN, ...); it is thus easy for the translator to "go through the places to translate".

An attempt was made to combine this function with automatic translation services but the result turned out to have a source layout that needed re-organization and thus was not acceptable to authors.

Zap Duplicates A tool to go through all the elements with an `id` attribute and make sure they are unique and, if not, de-duplicating them by adding a prefix to them. This was sometimes necessary after a refactoring method caused duplicate IDs which could only get ignored by the content storage (but were reported about by the storage reload).

The author is then left to adjust references to the elements, a non-trivial task that requires understanding of the references.

Appendix C

Glossary

absolutized : A reference from one element of content to another is called absolutized if said element can be taken out of it's original context and the reference still holds. In the case of the OMDoc files of ActiveMath, an absolutized reference is in the form `mbase://collection/theory/name`.
See Section 4.4.2.

API : An Application Programming Interface is the specification of a set of programmatic accesses that define a contract that enables the developers of a piece of software to rely on other pieces of software.
ActiveMath uses multiple Java* and JavaScript* APIs.

applet : An applet is generally understood as a Java applet: a part of a web-page which makes a visible rectangle allocated to a programme written in Java to perform multiple tasks. Java Applets are widely available on contemporary browsers although they tend to be slow to load and stop; they did not make their way into the emerging mobile browsers world.
ActiveMath uses Java applets for the function plotter (Section 2.5.5) the Wiris Input Editor (Section 2.4.6).

Confluence : A commercial Wiki web-server distributed by the Atlassian corporation; see `http://www.atlassian.com/software/confluence/`.

Connexions : A widely used authoring commons started for electrical engineering educational material based on XML* and MathML*-content. See `http://cnx.org/`, [Sch], [HB04], and [PNKJ08].

CSS : A standard of the world-wide-consortium to attribute style to elements of a document. CSS is widely used and implemented in many web-browsers – see [BLLJ98].

ActiveMath uses CSS intensively in all delivered web pages and in mathematical formulæ when present in HTML*.

Curriki A widely used learning objects repository for multiple formats of documents intended for school and university levels. Curriki is based on XWiki and is the base software of the i2geo* platform. See `http://www.curriki.org/` and [Kur08].

DFSL : Die Deutsche Freie Software Lizenz (the German Free Software License) is the software license chosen by the ActiveMath group as an equivalent of the GPL* adapted to the precise concepts of the German legal system among others the warranty terms which are not fully excludable in Germany. The DFSL has been written by Till Jäger and Till Kreutzer, see `http://www.d-fsl.de`.

DreamWeaver : An HTML editor used by the majority of web-designers. It is well known for its effective ability to edit in parallel code and rendering, thus offering good control of the code quality linked to a good display. See `http://www.macromedia.com/products/dreamweaver/`.

DTD : The Document Type Definition is a set of rules defining the content structure of XML* files; it is generally packaged in external files which tend to be recognized as the *specification* of the XML files; the DTDs were specified together with the XML format [BPSM97]. DTDs can define entities and implied attribute values (i.e. default values) which allow the XML to be kept considerably more compact than the actual parsing result.
In the ActiveMath authoring practice described in this thesis, DTDs are carried with the content collections. When using the *Start a collection...** command of jEditOQMath, authors use the DTD bundled inside the software. The DTDs that are bundled in the sofware are at: `http://svn.activemath.org/AMauthoring/trunk/projects/OQMathJEditPlugin/src/BaseContent/MineCollection/dtd/`.

Double Metaphone : An algorithm that converts words to *phonetic translations* with the property that two words that sound the same have the same phonetic translation. Double Metaphone is an algorithm of Lawrence Philips and is a refinement of Metaphone and Soundex. For more information, see the Wikipedia entry `http://en.wikipedia.org/wiki/Double_Metaphone`.
ActiveMath uses this tokenization mechanism to offer phonetic matching.

Dublin-Core : A standard to encode bibliographical information in XML* and RDF-based formats. See http://purl.org/DC.
ActiveMath uses the Dublin-Core field-names inside the metadata (Section 2.2.2).

ExaMat : The exercise authoring tool bundled in ActiveMath, described in [GT07]; see 3.3.5.

Flash : A virtual machine and programming environment for interactive content displayed in web-browsers, originally created by the MacroMedia corporation. Flash is often considered to be the most widely spread technology for rich content on the web. Nowadays, Flash has a single implementation distributed and maintained by the Adobe corporation. Its spread on the emerging mobile web platforms has been stopped.

FTP : The *File Transfer Protocol* is a widely used protocol to specify access of a client to a server to transfer individual files. See http://tools.ietf.org/html/rfc3659.

git : Modern versioning system offering strong decentralization, whereby each checkout can also become a repository of which other checkouts are made. See http://git-scm.com/.

GPL : The *General Public License*, the *GNU GPL*, is the main license of the Free Software Foundation. It is a *viral* license which imposes to a recipient of the license that any work *based on* the work he receives can only be redistributed under the same license. It is the license of major softwares such as Linux, Emacs, or MySQL. See http://www.gnu.org/licenses/gpl.html.

HTML : The *HyperText Markup Language* is the language that web-browsers most use to display web-pages. It describes a structure of sections and paragraphs, creating within the browser a document object model. A cousin of HTML, **XHTML**, does the same work as HTML but using an XML* syntax. The upcoming version of HTML, called HTML5, mostly describes a set of JavaScript* API* s that allow web-page authors to script highly interactive web-pages.
See http://www.w3.org/html/.

HTTP : The *HyperText Transfer Protocol* is used by web-browsers to communicate to web-servers; nowadays used by many other software pairs within the *web-services*. See http://www.w3.org/Protocols/rfc2616/rfc2616.html.

i2geo : A European project focussed on the sharing activities of interactive geometry, from 2007 to 2010. It has defined a common file format among the interactive geometry systems and has established a web-based platform to share the resources with previews, comments, and quality assessments across multiple languages and educational regions. See [LKM09].

Java : A programming language and family of virtual machines originated by the Sun MicroSystems corporation; Java appeared in the nineties as the *language of the internet*: it was both able to work for servers and within web-browsers, embedded in a *applet** – though the latter have suffered due to their limited speed. Java is, nowadays, widely used in entreprise and learning softwares; it is well supported by development environments. See `http://java.com`.
ActiveMath's server is programmed in Java as are most of the components of the authoring tools.

JavaScript : A programming language for execution within web-browsers in the context of web-pages. JavaScript was originally inspired by Java* but diverges in many respects (flexibility, functional orientation, lack of threads, less structured object orientation). Although an era existed where JavaScript was considered a threat and it was common to disable it on untrusted web-pages, more and more web sites require it for basic functions. ActiveMath, just as any web-application, carries its lot of JavaScript based on the jQuery library. JavaScript is specified by the ECMA consortium (`http://www.ecma-international.org//`) with API* standardized, among others, by the World Wide Consortium, e.g. in HTML*.

JNLP : The *Java Network Launch Protocol* is a specification of the web-based deployment of applications which can be run, in safety-nets or freely, on client computers. JNLP's most visible implementation is called Java Web Start, a trademark of Sun MicroSystems. Compared to applet* s, JNLP allows fully standalone applications, and cares better for code-signing, file-type associations, and complete code-caching.

ActiveMath uses JNLP to launch the assembly tool (Section 2.5.2) and concept-mapping tool (Section 2.5.3). Authors generally launch jEditOQMath using JNLP as well (see Section 3.5.1).

IMS-LTI : *Learning Tools Interoperability* is an emerging specification aimed at expressing documents that can be exchanged between two web-based learning environments, so that one can send the learner's browser to a different environment and back again for the purposes of using a tool – for example running an exercise. See `http://www.imsglobal.org/toolsinteroperability2.cfm`.

IMS-CP : *IMS Content Packaging* is a packaging standard for learning materials within a zip archive catalogued by a *manifest file* describing the role and metadata of each file and their organizations. IMS CP is the packaging means of SCORM*.

JDOM : This is an open-source library to manipulate in-memory representations of XML* documents in a way that bases itself on the Java collections; since its inception, JDOM aims at making the processing of XML documents easy to learn; this is opposed to SAX* and many other XML processing libraries for which the ease of learning by inexperienced developers has been a common challenge. See `http://www.jdom.org/`.
ActiveMath uses JDOM intensively, among others inside the storage (Chapter 4) and in the presentation pipeline (Section 2.4.7).

LeAM_calculus : One of the major content collections available for ActiveMath. See Section 9.3.2 and [LG06].

LOM : The Learning Object Metadata is an IEEE and IMS standard to annotate learning resources. It is generally considered applicable in environments such as the learning object repositories. LOM allows the framed construction of *application profiles* which allow specialization, generalization, enlargement, and restriction of the data schema. This often results in very diverse metadata standards whose interoperability is low.
ActiveMath's metadata is influenced in part by LOM, at least for the names, see Section 2.2.2.

Lucene : Apache Lucene is a widely used open-source information retrieval library of the Apache Foundation. It is central to many large scale search engines and is used by ActiveMath, both for storage (Chapter 4) and for search (Chapter 5). See `http://lucene.apache.org/` and [HG04].

media-type : The media-type, previously called *MIME-type*, is a string of ASCII characters denoting the type of a document, e.g. the content of a resource on the web, an attachment of an email, or file on the desktop. Media-types are specified by the IETF on an open basis similar to the RFC process, see `http://www.iana.org/assignments/media-types/index.html` for a list of existing assignments. ActiveMath uses media-types in particular HTTP* headers of the copy-and-paste operations (see Section 7.4) as well as in the delivery of content to web-browsers.
To date, media-types are not used for clipboard fragments. MacOSX uses the Uniform Type Identifiers which are similar but specify inheritance, while Windows allows almost arbitrary strings to be registered by applications.

MediaWiki : The Wiki system used by the Wikipedia encyclopedia project and supported by the WikiMedia foundation – see `http://www.mediawiki.org/`. MediaWiki is one of the example authoring tools in Section 1.3 and following.

metadata : *data about data*. In content management approaches, metadata denotes the information that is not part of the content but allows it to be structured. Metadata is often understood to contain fields of the Dublin-Core* standard. In the case of ActiveMath content items, see Section 2.2.2, the metadata includes most of the ingredients needed to create a user-model, serve course-generation, and display appropriately. This thesis presents ways to help the author to manage the metadata in Chapter 8.

MathML : The *Mathematical Markup Language* is a standard to encode mathematical formulæ by their presentations and their semantics. See [CIM10] for the latest MathML specification which includes a semantic markup equivalent to OpenMath*. MathML-presentation, that encodes the formula by their notation, is implemented in almost all contemporary browsers.
ActiveMath allows users to switch to the usage of MathML for formulæ which offers a higher quality of rendering but is less tested.

MOT : *My Online Teacher* is an authoring tool for adaptive systems aiming at authoring for, at least, AHA! and Whurle. See [CS06].

MPL : The *Mozilla Public License* was the license agreed upon when Netscape Inc. decided in 1998 to open-source its browser. This license imposes to recipients of such a license that any redistribution of the same files should be done under the same license but does not impose a requirement of the license on the other files. The MPL also provides clauses for disclosures of patent relevant to the received works. See `http://www.mozilla.org/MPL/`.

The MPL is the license of several components in jEditOQMath.

OpenMath : A standard describing the mathematical formulæ semantically in an XML* format. See Section 2.1.

PDF : The Portable Document Format is a specification of Adobe Systems Inc. meant as a followup of the PostScript page layout language. Together they provide a complete format for *e-paper*: a transmittable and fully predictable layout; they guarantee that a user receiving it, if he or she can open it, will see exactly the same thing. Moreover, PDF allows hyperlinks, searchability, table-of-contents, and many other features. PDF is commonly used for mathematical learning content (e.g. in exercise sheets). The main drawback of PDF lies in its very nature: the e-paper paradigm puts too great constraints on the consuming environment; mismatches commonly occur such as a text-line being too long for the consuming screen. HTML* avoids much of these issues by accepting an amount of impredictibility such as hyphenations, lack of font, or imprecise positionning.
ActiveMath's print service delivers PDF that is generated from TeX*.

properties : the *properties files* are text-files made of key-value pairs in the form `name = value`. The properties files are in common use in configuration and internationalization files in Java* and other platforms.

Authors generally edit properties file to adjust the configuration of their ActiveMath and, more importantly, to adjust the content-descriptor. See Section 4.4.1 and Section 6.4.

OWL : The *Web Ontology Language* is a W3C standard to encode ontologies made of classes, instances, relationships, sub-classes, and various other annotations. OWL extends the Resource Description Framework. See `http://w3.org/owl`.

SAX : The Simple API* for XML*-parsing is a set of methods that applications can implement so that a SAX-parser tells them the event of parsing the XML-documents. This form of parsing is deemed very fast because it does not consume memory but requires a conceptual inversion which makes it often hard to implement. See `http://sax.sourceforge.net/`.
SAX is used in ActiveMath's and jEditOQMath's parsing, mostly behind JDOM* or XSLT*.

schema : The term schema is often understood to be an XML-schema which describes the grammar of XML* documents. See `http://www.w3.org/standards/xml/schema`.
It can also be used to denote the information organization of a collection, for example the schema of a database.

SCORM : This standard is a well-established learning contents packaging system which is a zip file made of a few *organizations* referring to resources in the package expected to be browser visible; the zip file is organized around a manifest following the IMS-CP* packaging. SCORM also defines an API* for interactive exercises to report their scores. See [DT06]

TrackBack : A protocol to report to a web page (more precisely to the host of the page) that a link has been realized; typical of blog-postings talking about other blog-postings. See `http://www.movabletype.org/trackback/`. Trackback is often disabled on blogging platforms because it can be used to post links to spam; this has been the case of `http://eds.activemath.org/` for example (Section 9.2.4).

TeX : The TeX system is a software programme written by Donald Knuth in the seventies aimed at processing a source text into a quality layout for the print medium. Descendents of this software are used daily by most mathematicians of the university level, the dominant one being `pdflatex` which outputs PDF* from the LaTeX form.

TeX, generally written $\TeX$, is one of the example authoring tools in Section 1.3.

Unicode : A consortium and a standard cataloguing of the characters used in all possible writings on the earth, see `http://www.unicode.org`. The Unicode consortium also gathers other writing traditions such as the numeric patterns, the date-formats, or ways of noting the money.

ActiveMath exploits Unicode to deliver mathematical symbols to browsers.

jEditOQMath uses Unicode to allow the author to input mathematical symbols and provides, for them to be input, a set of abbreviations, see Section 3.4.2.

Velocity : Apache Velocity is a widely used open-source library for writing HTML*-presentation generators. ActiveMath uses it. Its constrained possibilities force the programmers to avoid difficult code writings in the view part of the Model-View-Controller pattern. See `http://velocity.apache.org/`.

Web-robot : A web-robot, also called *web-spider* or *web-crawler* is a software piece that pulls web-pages from web-servers, as if it was a normal browser, processes it and follows the links contained in the documents. Web-robots typically are used in search engines where the processing is an indexing process which is used to populate the index of the web-pages that are used in search results. Web-robots reach the *publicly accessible* web-documents delivered by the web-server that can be obtained by a sequence of clicks through the links. Being accessible by web-robots is generally a fundamental characteristic of a website. ActiveMath servers are web-robots accessible as explained in Section 5.3.1.

WYSIWYG *What You See Is What You Get*: an editing paradigm that stipulates that the edition process happens in a view that is equivalent to the view of the intended result.

See Section 1.2.1 for a critique of this paradigm.

WYCIWYG : *What You Check Is What You Get*: the response to WYSIWYG* formulated in this thesis to characterize the authoring approach whose result is obtained by various means but is *checked* within a target environment (verified, proofed, ...).
See Section 3.6.

XHTML+MathML : is the combination of the XML*-ized version of HTML*, called XHTML [Pem02], with the mathematical markup language MathML*. This combination is often used to use the abilities to render MathML in web-browsers although the newest HTML flavour, called HTML5, is embedding it in a non-XML environment.
ActiveMath's MathML* output uses this.

XML : The eXtensible Markup Language is a format for documents organized in a tree form where each branch is bounded by a start- and end-tag. Each tag can have atttributes, and have textual content. The grammar of XML documents are sometimes specified in DTD* s. Recommended in 1997, XML is a W3C standard, see [BPSM97].

XML-RPC : A simple HTTP* based protocol to achieve remote procedure calls. It has been created by Dave Winer long before the *competing standard* SOAP appeared. Its simplicity makes it widely implemented. See `http://www.xmlrpc.com/`. In ActiveMath and jEditOQMath, XML-RPC is used for the remote access to the content.

XSLT : The XML* StyLesheet Transformation language is a programming language to transform XML* trees into other XML trees, or into arbitrary textual sources. XSLT stylesheets are based on templates which are invoked either by name or when the tree matches a given pattern. The XSLT language is focussed on the processing of the documents and meant to have no external effects except delivering an output. It is widely implemented: multiple libraries are available; ActiveMath uses the Saxon library. Partial implementations are available within web-browsers.
See `http://www.w3.org/Style/XSL/`.

Bibliography

[AFNW07] Serge Autexier, Armin Fiedler, Thomas Neumann, and Marc Wagner. Supporting user-defined notations when integrating scientific text-editors with proof assistance systems. In *Towards Mechanized Mathematical Assistants - Proceedings of MKM 2007*, pages 176–190, 2007. See http://www.springerlink.com/content/b11778620u062466.

[AGC$^+$04] Andrea Asperti, Ferrucio Guidi, Claudio Sacerdoti Coen, Enrico Tassi, and Stefano Zacchiroli. A content based mathematical search engine: Whelp. In Jean-Christophe Filliatre, Christine Paulin-Mohring, and Benjamin Werner, editors, *Proceedings of the TYPES 2004 International Conference*, volume 3839 of *LNCS*, pages 17–32. Springer-Verlag, 2004. See also http://www.cs.UniBo.it/helm/.

[AMG$^+$03] Shaaron Ainsworth, Nigel Major, Shirley Grimshaw, Mary Hayes, Jean Underwood, Ben Williams, and David Wood. REDEEM: Simple intelligent tutoring systems from usable tools. In Murray et al. [MBA03].

[AMSK06] Vincent Aleven, Bruce M. McLaren, Jonathan Sewall, and Kenneth R. Koedinger. The Cognitive Tutor Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains. In Ikeda et al. [IAC06], pages 61–70. Available from http://ctat.pact.cs.cmu.edu/index.php?id=pubs.

[AMSK09] Vincent Aleven, Bruce McLaren, Jonathan Sewall, and Ken Koedinger. A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education*, 20, 2009.

[Atl06] Atlassian Inc. Remote api specification – confluence, 2006. Accessible at http://confluence.atlassian.com/display/DOC/Remote+API+Specification.

[AY07] Moody Altamimi and Abdou Youssef. A more canonical form of content mathml to facilitate math search. In *Proceedings of Extreme Markup Languages*, 2007. See `http://www.seas.gwu.edu/~ayoussef/publications.html`.

[BCC$^+$04] Stephen Buswell, Olga Caprotti, David Carlisle, Mike Dewar, Marc Gaëtano, and Michael Kohlhase. The OpenMath Standard, version 2.0. Technical report, The OpenMath Society, June 2004. Available at `http://www.openmath.org/`.

[BDFI06] Emanuela Busetti, Giuliana Dettori, Paola Forcheri, and Maria Grazia Ierardi. Promoting teachers' collaborative re-use of educational materials. In *Innovative Approaches for Learning and Knowledge Sharing, Proceedings of ECTEL-06*, 2006. See `http://www.springerlink.com/content/n25p524n4514/`.

[Ben06] Yochai Benkler. *The Wealth of Networks*. Yale University Press, 2006. See `http://www.benkler.org/wealth_of_networks/`.

[BL00] Michel Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-wimp user interfaces. In *Computer Human Interaction (CHI) 2002*, pages 446–453, 2000.

[BLLJ98] Bert Bos, Håkon Lie, Chris Lilley, and Ian Jacobs. Cascading Style Sheets, level 2 CSS2 Specification. W3C Recommendation. Technical report, World Wide Web Consortium, 1998.

[BM06] Bernd Krieg Brückner and Achim Mahnke. Semantic interrelation and change management. In Kohlhase [Koh06], chapter 26.6. See `http://www.omdoc.org/`.

[BPSM97] Tim Bray, Jan Paoli, and Michael Sperberg-McQueen. Extensible Markup Language (XML). W3C Recommendation PR-xml-971208, World Wide Web Consortium, December 1997. Available at `http://www.w3.org/TR/PR-xml.html`.

[BSS$^+$07] Paul De Bra, Natalia Stash, David Smits, Cristobal Romero, and Sebastian Ventura. Authoring and Management Tools for Adaptive Educational Hypermedia Systems: The AHA! Case Study. In P. De Bra, P. Brusilovsky, and R. Conejo, editors, *Evolution of Teaching and Learning Paradigms in Intelligent Environment*, volume 62 of *Studies in Computational Intelligence (SCI)*, pages 285–308. Springer-Verlag Heidelberg, 2007. See `http://www.springerlink.com/content/n6u63l27346444x7/`.

[BSYC05] Peter Brusilovsky, Sergey Sosnovsky, Michael Yudelson, and Girish Chavan. Interactive authoring support for adaptive educational systems. In Chee-Kit Looi and Gord McCalla, editors, *Proceedings of the 12th International Conference on Artificial Intelligence* *in Education AIED-2005*, pages 96–103, Amsterdam, The Netherlands, 2005. IOS Press. Available from http://www.sis.pitt.edu/~peterb/papers.html.

[Cai04] Paul Cairns. Informalising formal mathematics. In Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors, *Proceedings of 3rd Int. Conf on Mathematical Knowledge Management*, volume 3119 of *LNCS*, pages 58–72. Springer-Verlag, 2004. Available at http://www.uclic.ucl.ac.uk/paul/research/MizarLSI.pdf.

[CdM03] Alexandra. Cristea and Arno de Mooij. LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators. In *Proceedings WWW'03 (The Twelfth International World Wide Web Conference), Budapest,Hungary*, 2003. Available from http://www.www2003.org/.

[CIM10] David Carlisle, Patrick Ion, and Robert Miner. Mathematical markup language, version 3.0. W3C Recommendation, October 2010. Available at http://www.w3.org/TR/MathML3/.

[CLM$^+$08] Pavlina Chikova, Katrina Leyking, Gunnar Martin, Peter Loos, and Kai Peifer. Authoring-management-prozesse in industrieunternehmen. In Loos et al. [LZC08], pages 43–61.

[CM02] Ruth Clark and Richard. Mayer. *e-Learning and the Science of Instruction*. Pfeiffer, San Francisco, 2002.

[CMD05] Kris Cardinaels, Michael Meire, and Erik Duval. Automating metadata generation: the simple indexing interface. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 548–556, New York, NY, USA, 2005. ACM Press. Available from http://www2005.org/cdrom/HTML/p427/FinalWWW2005.htm.

[Cre08] Creative Commons. Choose a license, Accessed in April 2008. See http://creativecommons.org/license/.

[CS06] Alexandra Cristea and Craig Stewart. Automatic authoring of adaptive educational hypermedia. In Zong Min Ma, editor, *Web-based Intelligent E-Learning Systems: Technologies and Applications*, pages 24–55. Information Science Publishing (IDEA group), 2006.

[CSBC07] Alexandra Cristea, Craig Stewart, Tim Brailsford, and Paul Cristea. Adaptive hypermedia system interoperability: a 'real world' evaluation. *Journal of Digital Information*, 8(3), 2007. Available from http://journals.tdl.org/jodi/article/view/235/192.

[Dah06] Ingo Dahn. A metadata profile to establish the context of small learning objects: The slicing book approach. *International Journal on E-Learning*, 5(1):59–66, 2006. Available from `http://www.editlib.org/p/21742`.

[Dav00] James H. Davenport. A Small OpenMath Type System. *ACM SIGSAM Bulletin*, 34(2):16–21, 2000. (see also `http://www.openmath.org/standard/sts.pdf`).

[DCF06] Myroslava O. Dzikovska, Charles Callaway, and Elaine Farrow. Tools for hierarchical annotation of typed dialogue. In *Proceedings of the the EACL Workshop on Knowledge and Reasoning for Language Processing*, Trento, Italy, April 2006.

[DH03] Anton N. Dragunov and Jonathan L. Herlocker. Designing intelligent and dynamic interfaces for communicating mathematics. In *Proceedings of the 8th international conference on Intelligent user interfaces*, IUI '03, pages 236–238, New York, NY, USA, 2003. ACM.

[Doo10] Ahmad Salim Doost. Enhancement of activemath's student model. Master's thesis, Saarland University, 2010. Available from `http://www.activemath.org/Research/AllPublications`.

[DT06] Philip Dodds and Schawn E Thropp. SCORM 2004, 3rd ed., 2006. See `http://www.adlnet.gov/`.

[DT08] Paul Drijvers and Luc Trouche. *From artifacts to instruments: a theoretical framework behind the orchestra metaphor*, volume 2, chapter Research on Technology and the Teaching and Learning of Mathematics, pages 363–392. Information Age, Charlotte, NC, USA, 2008. K Heid and G Blume (Eds).

[Fau07] Arndt Faulhaber. Building a new learner model for activemath combining transferable belief model and item response theory. Master's thesis, Saarland University, 2007. Available from `http://www.activemath.org/Research/AllPublications`.

[FK00] A. Franke and M. Kohlhase. MBASE: Representing mathematical knowledge in a relational data base. In F. Pfenning, editor, *Proc. 17th International Conference on Automated Deduction (CADE)*, Lecture Notes on Artificial Intelligence. Springer-Verlag, 2000.

[Fro96] David Frohlich. Direct manipulation and other lessons. In *Handbook of HCI: Second completely revised edition*, chapter 22, pages 463 – 488. Elsevier Science, Amsterdam, 1996.

[gJSBF$^+$00] The Ω group: Jörg Siekmann, Chris Benzmüller, Armin Fiedler, Andreas Franke, Helmut Horacek, Paul Libbrecht, Michael Kohlhase, Andreas Meier, Erica Melis, Martin Pollet, Volker Sorge, Carsten Ullrich, and Jürgen Zimmer. Adaptive Course Generation and Presentation. In P. Brusilovski, editor, *Proceedings of ITS-2000 workshop on Adaptive and Intelligent Web-Based Education Systems*, pages 54–61, Montreal, 2000.

[Gog09] G. Goguadze. Representation for interactive exercises. In Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, editors, *Proceedings of 8th International Conference on Mathematical Knowledge Management MKM 2009*, volume 5625 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 294–309. Springer, 2009.

[Gog11] George Goguadze. *Knowledge Representation for Intelligent Exercises*. PhD thesis, Saarland University, Saarbrücken, 2011.

[GP03] George Goguadze and Alberto González Palomo. Adapting mainstream editors for semantic authoring of mathematics. In *Mathematical Knowledge Management Symposium, Heriot-Watt University, Edinburgh, Scotland*, November 2003. Available from `http://www.activemath.org/~george/publications.xml`.

[GPM05] George Goguadze, Alberto González Palomo, and Erica Melis. Interactivity of Exercises in ActiveMath. In *In Proceedings of the 13th International Conference on Computers in Education (ICCE 2005)*, pages 107–113, Singapore, 2005.

[Gra96] Peter Graf. *Term Indexing*, volume 1053 of *LNCS*. Springer-Verlag, 1996.

[Gra07] Preston Gralla. U.S. senator: It's time to ban wikipedia in schools, libraries, February 2007. See `http://blogs.computerworld.com/node/4598`, accessed (Aug 2009).

[GT07] George Goguadze and Ian Tsigler. Authoring Interactive Exercises in ActiveMath. In *Proceedings of the MathUI Workshop at Mathematical Knowledge Management Conference*, Linz, Austria, June 2007.

[GT08a] Ghislaine Gueudet and Luc Trouche. Du travail documentaire des enseignants : genèses, collectifs, communautés. le cas des mathématiques. *Education et didactique*, 2(3):7–33, 2008.

[GT08b] Dominique Guin and Luc Trouche. Un assistant méthodologique pour étayer le travail documentaire des professeurs : le cédérom sfodem 2008. *Repères-IREM*, (72):5–24, 2008. See also `http://sfodem.um2.fr/`.

[GUM$^+$04] George Goguadze, Carsten Ullrich, Erica Melis, Jörg Siekmann, Christian Groß, and Rafael. LeActiveMath Structure and Metadata Model. Deliverable D6, LeActiveMath Consortium, 2004. Available from `http://www.leactivemath.org/publications1.html`.

[HB04] Geneva Henry and Richard Baraniuk. Peer to peer collaboration using connexions. In *American Society for Engineering Education Annual Conference & Exposition Salt Lake City, Utah*, June 2004. Available from `http://cnx.org/aboutus/publications`.

[HBLM02] Jim Hendler, Tim Berners-Lee, and Erik Miller. Integrating applications on the semantic web. *Journal of the Institute of Electrical Engineers of Japan*, 122(10):676–680, October 2002. See `http://www.w3.org/2002/07/swint`.

[Hea09] Marti A. Hearst. *Search User Interfaces*. Cambridge University Press, 2009. See `http://searchuserinterfaces.com/`.

[HG04] Erik Hatcher and Otis Gospodnetic. *Lucene in Action (In Action series)*. Manning Publications, December 2004.

[Hom06] Martin Homik. Assembly Tool. Deliverable D37, LeActiveMath Consortium, 2006. Available from `http://www.leactivemath.org/publications1.html`.

[HPRR10] Jónathan Heras, Vico Pascual, Ana Romero, and Julio Rubio. Integrating multiple sources to answer questions in algebraic topology. In Serge Autexier, , Jacques Calmet, David Delahaye, Patrick D. F. Ion, Laurence Rideau, Renaud Rioboo, and Alan P. Sexton, editors, *AISC/MKM/Calculemus*, volume 6167, pages 331–335. Springer Verlag, July 2010.

[IAC06] Mitsuru Ikeda, Kevin D. Ashley, and Tak-Wai Chan, editors. *Intelligent Tutoring Systems, 8th International Conference, ITS 2006, Jhongli, Taiwan, June 26-30, 2006, Proceedings*, volume 4053 of *Lecture Notes in Computer Science*. Springer, 2006.

[Jed09] Dominik Jednoralski. Latent semantic algorithms for improving search results in an intelligent tutoring environment for mathematics. Technical report, University of Saarland, 2009. Bachelor thesis.

[Jed10] Dominik Jednoralski. Detection of structural, linguistic and semantic content inconsistencies in an adaptive web-based learning environment. Master's thesis, University of Saarland, 2010.

[JW04] Ian Jacobs and Norman Walsh. Architecture of the world wide web. Technical report, World Wide Web Consortium, 2004. Available at `http://www.w3.org/TR/webarch/`.

[Koh00] Michael Kohlhase. OMDOC: Towards an OPENMATH representation of mathematical documents. Seki Report SR-00-02, Fachbereich Informatik, Universität des Saarlandes, 2000. See also `http://www.omdoc.org/`.

[Koh04] Michael Kohlhase. Semantic markup for TeX/LaTeX. In *Proceedings of MathUI'04, Bialowieza, Poland*, Sept 2004. See `http://www.activemath.org/~paul/MathUI04/proceedings/LTeX2OMDoc.html`.

[Koh06] Michael Kohlhase. *OMDoc: An Open Markup Format for Mathematical Documents [version 1.2]*, volume 4180/2006 of *LNCS*. Springer Verlag Heidelberg, 2006. See `http://www.omdoc.org/`.

[Kor99] Ulrich Kortenkamp. *Foundations of Dynamic Geometry*. PhD thesis, ETH Zürich, Zürich, 1999.

[Kra04] Milos Kravcik. Specfication of adaptation strategy by FOSP method. In Alexandra Cristea, editor, *Proceedings of the workshop Authoring of Adaptive and Adaptable Educational Hypermedia*, 2004. Available at `http://www.dcs.warwick.ac.uk/~acristea/AH04/papers/`.

[Kre07] Till Kreutzer. Rechtsfragen bei E-Learning: Ein Praxis-Leitfaden. Technical report, Multimedia Kontor, Hamburg, 2007. Available at `http://www.mmkh.de/upload/dokumente/Leitfaden_E-Learning_und_Recht_creativecommons_MMKH.pdf`.

[KSO04] Milos Kravcik, Marcus Specht, and Reinhard Oppermann. Evaluation of winds authoring environment. In *Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems, AH 2004, Eindhoven, LNCS 3137*, 2004.

[KUM06] Philipp Kärger, Carsten Ullrich, and Erica 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.

[Kur08] Barbara Kurshan. OER Models that Build a Culture of Collaboration: A Case Exemplified by Curriki. *E-learning Papers*, 10, 2008. Available at `http://www.elearningpapers.eu/index.php?lng=en&page=doc&doc_id=12406&doclng=6&vol=10`.

[LAG09] Paul Libbrecht, Eric Andrès, and Yecheng Gu. Smart Pasting for ActiveMath Authoring. In *Proceedings of MathUI'09*, July 2009. Available from `http://www.activemath.org/workshops/MathUI/09/`.

[Lan07] Christoph Lange. SWIM: A Semantic Wiki for Mathematical Knowledge Management. In *Proceedings of MathUI'07, Linz*, 2007. Available from `http://www.activemath.org/workshops/MathUI/07/`.

[LF03] P. Libbrecht and M. Fuchs. Interface MBaseRef. Java Object Documentation, 2003. `http://www.activemath.org/~ilo/redirs/MBaseRef.html`.

[LG06] Paul Libbrecht and Christian Groß. Experience report writing LeActiveMath Calculus. In Jon Borwein and William Farmer, editors, *Proceedings of Mathematical Knowledge Management 2006*, number 4108 in LNAI, pages 251–265. Springer Verlag, aug 2006. Available at `http://www.activemath.org/pubs/bi.php/Libbrecht-Gross-Experience-Report-Authoring-LeAM-MKM-2006`.

[Lib06] Paul Libbrecht. Enhanced Dictionary. Deliverable D15, LeActiveMath Consortium, 2006. See `http://www.leactivemath.org/publications1.html`.

[Lib07] Paul Libbrecht. Content dictionary notations. In *Proceedings of the OpenMath Workshop, Linz Austria, 2007*, 2007. See `http://jem-thematic.net/node/167`.

[Lib08] Paul Libbrecht. 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, Sept 2008.

[Lib10] Paul Libbrecht. What You Check is What You Get: Authoring with jEditOQMath. In *Proceedings 10th IEEE International Conference on Advanced Learning Technologies.*, pages 682–686. IEEE, july 2010.

[Lib11] Paul Libbrecht. Re-use? is this re-use? *ZDM*, 43:353–358, 2011. 10.1007/s11858-011-0336-3.

[LJ06] Paul Libbrecht and Dominik Jednoralski. Drag and Drop of Formulae from a Browser. In *Proceedings of MathUI'06*, August 2006. Available from `http://www.activemath.org/~paul/MathUI06/`.

[LKM09] Paul Libbrecht, Ulrich Kortenkamp, and Christian Mercat. I2Geo: a Web-Library of Interactive Geometry. In *Proceedings of DML 2009*, July 2009. See presentations at `http://www.fi.muni.cz/~sojka/dml-2009.xhtml`.

[LM06] Paul Libbrecht and Erica Melis. Methods for Access and Retrieval of Mathematical Content in ActiveMath. In Nobuki Takayama, Andres Iglesias, and Jaime Gutierrez, editors, *Proceedings of ICMS-2006*, number 4151 in LNCS. Springer Verlag GmbH, september 2006.

Available from `http://www.activemath.org/pubs/bi.php?id=Libbrecht-Melis-Access-and-Retrieval-ActiveMath-ICMS-2006`.

[Lok98] Matija Lokar. Derive v slovenskih srednjih šolah. Analiza anket, zavod za šolstvo, National Education Institute of Slovenia, 1998.

[LZC08] Peter Loos, Volker Zimmermann, and Pavlina Chikova, editors. *Prozessorientiertes Authoring Management*. Logos Verlag, Berlin, 2008.

[Mah06a] Kavi Mahesh. Text retrieval quality: A primer. Technical report, Oracle Coproration, 2006. See `http://www.oracle.com/technology/products/text/htdocs/imt_quality.htm`.

[MAH06b] T. Mossakowski, S. Autexier, and D. Hutter. Development graphs – proof management for structured specifications. *Journal of Logic and Algebraic Programming*, 67(1-2):114–145, 2006.

[Man05] Shahid Manzoor. Authoring presentation semantics for mathematical documents for the web. Master's thesis, University of Saarland, 2005. Available from `http://www.referent-tracking.com/RTU/?page=manzoor_vita.phtml&basepath=..%2Fshahid%2F`.

[MBA03] Tom Murray, Stephen Blessing, and Sharon Ainsworth. *Authoring Tools for Advanced Technology Learning Environment*. Kluwer Academic Publishers, Dordrecht, 2003.

[MBG$^+$03] E. Melis, J. Büdenbender, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. Knowledge representation and management in activemath. *Annals of Mathematics and Artificial Intelligence, Special Issue on Management of Mathematical Knowledge*, 38(1-3):47–64, 2003. Volume is accessible from `http://monet.nag.co.uk/mkm/amai/index.html`.

[McM04] Flora McMartin. Merlot: A model of user involvement in digital library design and implementation (a case study). *Journal of Digital Information*, 05(03), September 2004. See `http://jodi.ecs.soton.ac.uk/Articles/v05/i03/McMartin/`.

[MEC$^+$06] Daniel Marquès, Ramon Eixarch, Glória Casanellas, Bruno Martïñez, and Tim Smith. WIRIS OM Tools a Semantic Formula Editor. In *Proceedings of MathUI'06*, August 2006. Available from `http://www.activemath.org/~paul/MathUI06/`.

[MGH$^+$06] Erica Melis, George Goguadze, Martin Homik, Paul Libbrecht, Carsten Ullrich, and Stefan Winterstein. Semantic-Aware Components and Services of ActiveMath. *British Journal of Educational Technology*, 37(3):405–423, may 2006.

[MGLU09a] Erica Melis, George Goguadze, Paul Libbrecht, and Carsten Ullrich. ActiveMath - A Learning Platform With Semantic Web Features. In D. Dicheva, R. Mizoguchi, and J. Greer, editors, *Semantic Web Technologies for e-Learning*, volume 4, pages 159–177. IOS Press, 2009.

[MGLU09b] Erica Melis, George Goguadze, Paul Libbrecht, and Carsten Ullrich. Culturally aware mathematics education technology. In Emmanual Blanchard and Daniéle Allard, editors, *The Handbook of Research in Culturally-Aware Information Technology: Perspectives and Models*, pages –. IGI-Global, 2009.

[MHRS06] Marek Meyer, Tomas Hildebrandt, Christoph Rensing, and Ralf Steinmetz. Requirements and an architecture for a multimedia content re-purposing framework. In *EC-TEL*, pages 500–505, 2006.

[MKH05] Erica Melis, Philipp Kärger, and Martin Homik. Interactive Concept Mapping in ActiveMath (iCMap). In Jörg M. Haake, Ulrich Lucke, and Djamshid Tavangarian, editors, *Delfi 2005: 3. Deutsche eLearning Fachtagung Informatik*, volume 66 of *LNI*, pages 247–258, Rostock, Germany, sep 2005. Gesellschaft für Informatik e.V. (GI).

[ML71] Saunders Mac Lane. *Categories for the working mathematician*. Springer-Verlag, New York, 1971.

[MLUM06] Shahid Manzoor, Paul Libbrecht, Carsten Ullrich, and Erica Melis. Authoring presentation for OpenMath. In Michael Kohlhase, editor, *Mathematical Knowledge Management: 4th International Conference, MKM 2005, Bremen, Germany, July 15-17, 2005, Revised Selected Papers*, volume 3863 of *LNCS*, pages 33–48, Heidelberg, 2006. Springer.

[MM07] Robert Miner and Rajesh Munavalli. An approach to mathematical search through query formulation and data normalization. In Manuel Knauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Towards Mechanized Mathematical Assistants - Proceedings of MKM 2007*, pages 342–355. Springer Verlag, Berlin, Germany, July 2007.

[MMU$^+$07] Erica Melis, Marianne Moormann, Carsten Ullrich, George Goguadze, and Paul Libbrecht. How activemath supports moderate constructivist mathematics teaching. In *8th International Conference on Technology in Mathematics Teaching*, Hradec Kralove, 2007.

[Mon05] Adolfo Montalvo. Conceptual model for odl quality process and evaluation grid, criteria and indicators. Technical report, e-Quality Project, 2005. See `http://www.e-quality-eu.org/`.

[MSK01] M. Murata, S. St.Laurent, and D. Kohn. XML media types, January 2001. `ftp://ftp.rfc-editor.org/in-notes/rfc3023.txt`.

[MTD$^+$06] David Millard, Feng Tao, Karl Doody, Arouna Woukeu, and Hugh Davis. The knowledge life cycle for e-learning. *Int. J. Cont. Engineering Education and Lifelong Learning*, 16(1/2):110–121, 2006. Available at `http://eprints.ecs.soton.ac.uk/11906/`.

[Mur03] Tom Murray. Principles for pedagogy-oriented knowledge-based tutor authoring systems: lessons learned and a design meta-model. In Murray et al. [MBA03].

[MvLB06] Rafael Morales, Nicolas van Labeke, and Paul Brna. Approximate modelling of the multi-dimensional learner. In Ikeda et al. [IAC06], pages 555–564.

[NBM07] Nicolas van Labeke, Paul Brna, and Rafael Morales. Opening up the interpreation process in an open learner model. *International Journal of Artificial Intelligence in Education*, 17(3):305–338, August 2007.

[ND02] F. Neven and E. Duval. Reusable learning objects: a survey of LOM-based repositories. In *Proceedings of the 10th ACM International Conference on Multimedia*, pages 291–294, 2002.

[Pal06] Alberto González Palomo. QMath: A human-oriented language and batch formatter for OMDoc. In Kohlhase [Koh06], chapter 26.2. See `http://www.omdoc.org/`.

[Pem02] Steven Pemberton. XHTML^{TM} 1.0 The Extensible HyperText Markup Language (Second Edition). W3c recommendation, World Wide Web Consortium, Available at `http://www.w3.org/TR/xhtml1.`, Jan 2002.

[PNKJ08] Lisa Petrides, Lilly Nguyen, Anastasia Kargliani, and Cynthia Jimes. Open educational resources: Inquiring into author reuse behaviors. In *Proceedings of ECTEL 2008, Maastricht, Markus Specht and Pierre Dillenbourg (eds)*, volume 5192 of *LNCS*, pages 344–353. Springer Verlag, Sept 2008.

[Por80] Martin Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980. See also `http://tartarus.org/~martin/PorterStemmer/`.

[Rab07] Florian Rabe. OMDoc Theory Graphs Revisited. In *Proceedings of the OpenMath/JEM workshop*, 2007. Available from `http://www.openmath.org/meetings/linz2007/`.

[RB03] Pierre Rabardel and Gaëtan Bourmaud. From computer to instrument system: a developmental perspective. *Interacting with Computers*, 15(5):665 – 691, 2003. Part 1 Organizational Issues.

[RCM06] Robby Robson, Geoff Collier, and Brandon Muramatsu. Reusability framework. Technical report, Reusable Learning Project, 2006. See `http://www.reusablelearning.org/`.

[RPA+09] Leena Razzaq, Jozsef Parvarczki, Shane Almeida, Manasi Vartak, Mingyu Feng, Neil Heffernan, and Kenneth Koedinger. The assistment builder: Supporting the life-cycle of its content creation. *IEEE Transactions on Learning Technologies Special Issue on Real-World Applications of Intelligent Tutoring Systems*, 2(2):157–166, 2009.

[RSG08] Devshri Roy, Sudeshna Sarkar, and Sujoy Ghose. Automatic extraction of pedagogic metadata from learning content. *Int. J. Artif. Intell. Ed.*, 18:97–118, April 2008.

[RZM+08] Christoph Rensing, Birgit Zimmermann, Marek Mayer, Lasse Lehmann, and Ralf Steinmetz. Wiederverwendung von multimedialen Lernressourcen im Re-Purposing und Authoring by Aggregation. In Loos et al. [LZC08], pages 19–40.

[Sch] Philip Schatz. Math editor introduction. Available at `http://cnx.org/content/m24561/latest/`; retrieved on May 8 2010.

[Sch07] Rolf Schulmeister. *Grundlagen hypermedialer Lernsysteme: Theorie - Didaktik - Design*. Oldenbourg, 2007.

[Sha99] Mike Sharples. *How we write: writing as creative design*. Routledge, London, New York, 1999.

[Shn83] Ben Shneiderman. Direct manipulation: a step beyond programming languages. *IEEE Computer*, 16(8):57–69, August 1983.

[SK06] Joan Sucan and Michael Kohlhase. A search engine for mathematical formulae. In Tetsuo Ida, Jacques Calmet, and H. Wang, editors, *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2006*, number 4120 in LNAI. Springer Verlag, 2006. See also `http://kwarc.eecs.iu-bremen.de/software/mmlsearch/`.

[SL11] Petr Sojka and Martin Líska. Indexing and searching mathematics in digital libraries - architecture, design and scalability issues. In James H. Davenport, William M. Farmer, Josef Urban, and Florian Rabe, editors, *Intelligent Computer Mathematics - Proceedings*, number 6824 in LNCS, pages 228–243, 2011.

[SMMMG06] Hala Skaf-Molli, Pascal Molli, Olivera Marjanovic, and Claude Godart. LibreSource : Web Based Platform for Supporting Collaborative Activities. In *2nd IEEE Conference on Information and Communication Technologies: from Theory to Applications*, April 2006.

[Sos10] Sergey Sosnovsky. Metadata cookbook. Technical Report Supplement to D1.1, Math-Bridge Consortium, June 2010.

[SPH+07] Tim Smith, Helen Pain, Ruth Hanson, Jeff Haywood, and Hamish Macleod. Test results college/university. Technical Report D44, The LeActiveMath Consortium, December 2007.

[SS06] Alan P. Sexton and Volker Sorge. Abstract matrices in symbolic computation. In Barry M. Trager, editor, *Symbolic and Algebraic Computation, International Symposium, ISSAC 2006, Genoa, Italy, July 9-12, 2006, Proceedings*, pages 318–325. ACM, 2006.

[TBN09] Jana Trgalova, Denis Bouhineau, and Jean-François Nicaud. An Analysis of Interactive Learning Environments for Arithmetic and Algebra Through an Integrative Perspective. *International Journal of Computers for Mathematical Learning*, 14 (3):299–331, 2009.

[TSW05] Martin Theobald, Ralf Schenkel, and Gerhard Weikum. An efficient and versatile query engine for topx search. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per ?ke Larson, and Beng Chin Ooi, editors, *VLDB*, pages 625–636. ACM, 2005.

[TWS04] Martin Theobald, Gerhard Weikum, and Ralf Schenkel. Top-k query evaluation with probabilistic guarantees. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *VLDB*, pages 648–659, 2004. Available at `http://www.vldb.org/conf/2004/RS17P3.PDF`.

[Ull07] Carsten Ullrich. *Course Generation as a Hierarchical Task Network Planning Problem*. PhD thesis, Saarland University, Saarbrücken, 2007. Available from `http://www.carstenullrich.net/`.

[ULWM04] Carsten Ullrich, Paul Libbrecht, Stefan Winterstein, and Martin Mühlenbrock. A flexible and efficient presentation-architecture for adaptive hypermedia: Description and technical evaluation. In Kinshuk, C. Looi, E. Sutinen, D. Sampson, I. Aedo, L. Uden, and E. Kähkönen, editors, *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies (ICALT 2004), Joensuu, Finland*, pages 21–25, 2004.

[Urb04] Josef Urban. MoMM - Fast Interreduction and Retrieval in Large Libraries of Formalized Mathematics. In *Proceedings of the ESFOR 2004 workshop at IJCAR'04 available at http://www.mpi-sb.mpg.de/~baumgart/ijcar-workshops/proceedings/PDFs/WS5-final.pdf.*, 2004. More information on the project at `http://wiki.mizar.org/cgi-bin/twiki/view/Mizar/MoMM`.

[vR79] Cornelis Joost van Rijsbergen. *Information Retrieval*. Butterworths, 1979. Available at `http://www.dcs.gla.ac.uk/~iain/keith/`.

[You05] Abdou Youssef. Information search and retrieval of mathematical contents: Issues and methods. In *proceedings of the ISCA 14th International Conference on Intelligent and Adaptive Systems and Software Engineering (IASSE-2005)*, Toronto, Canada, 2005.

[Zim08] Volker Zimmermann. Vom Stand-Alone-Autorenwerkzeug zur prozessorientierten Autorenplattform: Grundidee und Lösungsansatz. In Loos et al. [LZC08], pages 43–61.

Index