What You Check is What You Get: Authoring with jEditOQMath

Paul Libbrecht
Competence Center for E-Learning, DFKI and University of Saarland, Saarbrücken, Germany
http://www.activemath.org/~paul/

***Abstract*—jEditOQMath is an authoring tool for the intelligent learning environment ActiveMath. Its editing interface is a simple source editor. However the wealth and power of the authoring tool lies elsewhere, namely in its integration into the learning platform, so that authors design a great learning experience and not simply a great content editing view. In this paper, we describe the tool and propose to name its paradigm WYCIWYG: What you Check is What You Get. The instrumental approach of human computer interaction is applied to analyze the practice and to explain how the edited source becomes the instrument to communicate with the learning environment.**

***Keywords*-Technology enhanced Science Education; Learning Systems Platforms and Architectures; Intelligent Educational Systems**

I. Introduction

Authoring for E-learning is well known to require expertise from pedagogical, domain-specific, and technical levels. Moreover, the technical expertise needs to lie both in the usage of the environment the learners will encounter and in the usage of the authoring tools. Therefore, making authoring more accessible is an objective of research.

We focus on authoring for mathematics learning on the web: this adds the challenges of the mathematical formulæ (to be properly rendered, to interact properly with tools) to the traditional web-challenges (multiple methods of navigation and somewhat unpredictable clients).

The described tool is a tool for the content-creation for the ActiveMath learning environment [1]. It involves (1) editing OMDoc files [2] with their semantic mathematical nature and (2) experimenting with them in the learning environment. The authoring tool approach covers aspects (1) and (2) giving to the author significant opportuninities to platform experimentation, discovery, and practice.

A. Outline

This paper starts with an introduction of the learning environment, and a presentation of the related research and tools in authoring. It then describes the normal authoring set up, and shows the three interaction modalities of jEditOQMath: the files, the sources, the learning platform. It follows with an exploration of how the old concept of *direct manipulation* [3] or the more modern *instrumental approach* [11] can be applied in this case. An outlook concludes the paper.

B. The ActiveMath Learning Environment

The ActiveMath learning environment is an intelligent web-based learning environment for mathematics. It serves learning materials encoded in the OMDoc language [2] to web-browsers adapting to their capabilities: in HTML+CSS or in XHTML+MathML (see [4]). The semantic nature of the OMDoc content allows many usages of the mathematical formulæ, for example, to use them for copies in interactive exercises or to plot them. ActiveMath maintains a model of the learners' competencies which enables it to suggest learning content that best suits a given pedagogical objective. Several tools allow learners to experiment with the mathematical knowledge behind the content (such as the search tool or the interactive concept-map tool).[1]

The content of ActiveMath is encoded in OMDoc [2]. This language partitions content elements in mathematical items (e.g. a definition, an example), each with its own identifier. The items are annotated with metadata and the content is made up of text, hyperlinks, and formulæ encoded in the OpenMath semantic format. The language is rich in references: for pedagogical annotations (e.g. "this definition requires this theorem"), to allow hyperlinks ("you may also re-read the rule of integrals of sums"), or to allow extensible mathematical symbols in formulæ ("the set of natural numbers, the one with zero").

C. Related Works

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 PowerPoint, 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* paradigm. ActiveMath'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 [5]. Most of the descriptions there are descriptions of the knowledge models while we focus on the description of an authoring workflow.

[1]See [1] for more details as well as the software's web-site http://www.activemath.org/Software.

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 [6, 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 [7] 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.

II. Set up

As is described in jEditOQMath tutorial[2], we recommend that authors use three tools for authoring:

- an **author ActiveMath**: a server on the author's machine reading the files stored on the local computer
- a classical desktop **file-management** interface to manage files, grouped in content-collections (simple directories see [8]); the files are the *content sources*.
- a text **editor** presenting the content of the files and allowing them to be enriched and modified.

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

III. Tame Source Authoring with jEditOQMath

The activity proposed to authors follows the classical text-source-authoring practices as used by the TEX and MediaWiki: its input is provided by the editing of a text document and its exploitation is done in a separate preview space. The source editing, however, is not free plain text, which is far too free to provide any useful information to a system to deliver rich services. Similarly to others, it is structured along a basic grammar given by the OMDoc format. Moreover, many of its ingredients only make sense when exploited by the ActiveMath learning environment.

The author acts on each of the three levels defined above using commands to go from one to the other and back.

[2]The tutorial is made of steps in the book of tasks: http://eds.activemath.org/en/tasks. A video can be previewed there describing most of the features of this paper.

A. *Interactions with the files*

As is the case with most data collection, the materialization of the content of a project is made in directories of files that live on the desktop of the author. This allows for multiple preservation, archiving, and sharing mechanisms that are available for them: send by mail, download, synchronize, put aside, backup: those are all activities that are well known for files. Content files in ActiveMath are stored in directories, the content-collections, which form the unit of exchange as explained in [8].

Files are also the elements an author can use to reproduce his set up, for example uploading them to the ActiveMath server of the school.

B. *Interactions with the Sources*

The textual source files of jEditOQMath are in a format called OQMath: these files are XML files; they are *mostly* in the OMDoc format but the OpenMath objects are in a shorter syntax with $-separated islands. The editing of the XML syntax and the maintenance of its readability is possible because the grammar they obey to allows compact fragments with hidden values provided by the DTD.

The manipulation of the text source follows the classical editing practices of texts which include actions such as: copy-and-paste, simple keyboard input, drag-and-drop, search-and-replace. All of these gestures are widely known and implemented in an intuitive fashion in this classical editor, jEdit.[3] Characteristic of a user-friendly input syntax, the XML nature of the OQMath files is sufficiently flexible to provide sufficient freedom to cater for personal preferences in the placement of the tags; this freedom is known to be widely used by TEX users.

1) Mathematical Formulæ: The mathematical formulæ of the OQMath sources are meant to be processed by the QMath tool: a processor that allows a very short and customizable syntax to encode OpenMath objects. For example `$b^2-4ac$`, `$tan(π/2-x)=neg(tan(x-π/2))$` to encode $b^2 - 4ac$ and $\tan(\pi/2) = -\tan(x - \pi/2)$.

The customizability of the formulæ input is based on notation definitions that authors can extend. These notation definitions are the main input for the QMath [9] processor which jEditOQMath uses. The customizability, although detrimental to an easy exchange of fragments, seems to be a critical freedom which mathematicians constantly exploit. jEditOQMath supports the wide use of graphical symbols in mathematics by supporting Unicode as well as enabling the input through predefined *abbreviations*, which transform such words as `Delta` to the greek letter Δ: the user types the letters then requests the *expansion* which transforms the letters into the single visible character. This approach raises readability greatly compared to classical TEX sources, which keep macros as words.

[3]jEdit is a java-based editor under the GPL, see http://www.jedit.org/.

2) Populating the Source: jEditOQMath helps inputting "the right text".This is presented here while, in the next section, we shall see how to make sure it is "correct".

First-time users of jEditOQMath will start by using the templates, to follow the default practice. The templates allow to create the skeleton of a new document, all common-use content-items, and common constructs inside the items (such as formulæ or images). Because templates are made to create content that is modified thereafter, the expected places to be filled are marked with template-zones, islands surrounded by French quotes. After the template is inserted, a click jumps from template zone to template zone, by a command.

Copying from outside of jEditOQMath is another way to start inputting content. An area where input is challenging is the **input of references**: they are of utmost importance to construct the knowledge structure 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 (for example the title of an item in the rendering of a book-page in ActiveMath). Another way to insert references is to the "searchable items list", a tool to find items by IDs in order to navigate to or reference them.

Copy-and-paste of formulæ is another area where a transfer from outside is offered: some TEX-encoded and MATHML-encoded sources are supported. This paste function is of use to discover the encoding of mathematical formulæ from the web in the wild 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.

Finally, **input of the XML tags** that constitute the structure of the document is supported by the grammar. It has been a deliberate choice to keep the OQMath files in XML format: many tools with a rich feature set support its edition and this support is fundamental for users that are unsure of the possible inputs. jEdit, with its XML plugin, offers at the input of the start-tag character < a pop-up indicating all possible children (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, to fold them: this has been most often used by authors to maintain readability.

We have seen 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.

C. Validation

Having input the necessary text, the author wishes to make sure it is **understandable**, and **it is working as intended** since the source is only a means towards the creation of an anticipated learning experience. This fundamental expectation has to be checked by the author.

The first method to ensure it is understood is called validation: an automated process that performs checks on the content and reports to the user.

The first validation happens upon saving: it is checked that the XML is well-formed and DTD-grammar-compliant. The reported errors are presented in an error list organized by file and line numbers as predicted by the system.

Two more validations happen during the *build process* initiated by the author to transmit the content to the server.

The conversion of mathematical formulæ may trigger conversion errors which are reported similarly. Finally, an important validation is done at the end of the build process when the content storage resolves all references. The references are input in short forms (often as relative references) and the resolution uses `imports` elements to resolve them. Resolution errors are shown within the validation and QMath-transformation errors: Because the QMath transformation is tuned to map lines to lines, the reference error reports are presented at the right line number in the OQMath sources hence are, for most easy to fix in place.

D. Exploitation of the Content Sources: the Build

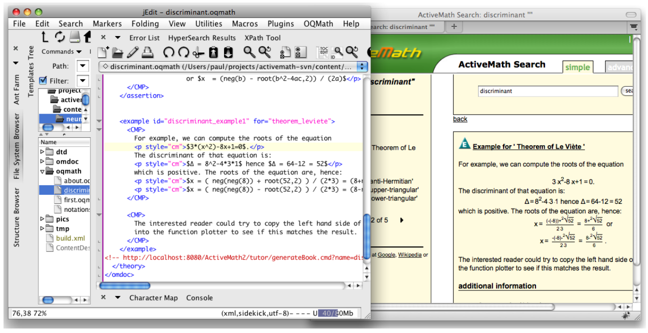
The *build process* is invoked by running an Ant build by the author who wishes to preview his content. The build process triggers: conversion of the QMath islands, the automated production of collection organizations (a table-of-contents as explained in [4]), a reload to the author's ActiveMath's content storage, which resolves all references, and invalidates the changed items of the presentation-system's cache (see [4]). The build process sketched here is a routine task for all.

Once the build process has succeeded, the author can view the content. If he has just been inputting new content he will be looking at its rendering by the navigation to the appropriate page. Quite often, however, the content is edited beyond the simple input of new items.

E. Interactions with the ActiveMath Platform

In the previous sections, we have 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 experience will be like. As often done on the web, most of the ActiveMath web-browser pages which deliver content can simply be reloaded to see the effect of the changes of content from the same perspective, e.g:

- in pages of a pre-recorded book, a click on the page link will display the updated content
- search queries are stored with the user-history, a reload will do the search again, on the updated data
- an interactive exercise window carries all the inputs in its URL; a reload will reproduce each user's interaction; this can be used, for example, to check feedback repairs


jEdit – discriminant.oqmath
File Edit Search Markers Folding View Utilities Macros Plugins OQMath Help
Error List HyperSearch Results XPath Tool
discriminant.oqmath (/Users/paul/projects/activemath-svn/content/…
Character Map Console
ActiveMath Search: discriminant ""
ActiveMath Search
simple
discriminant
back
additional information

```
            or $x  = (neg(b) - root(b^2-4ac,2)) / (2a)$</p>
    </CMP>
  </assertion>


  <example id="discriminant_example1" for="theorem_leviete">
    <CMP>
      For example, we can compute the roots of the equation
      <p style="cm">$3*(x^2)-8x+1=0$.</p>
      The discriminant of that equation is:
      <p style="cm">$Δ = 8^2-4*3*1$ hence $Δ = 64-12 = 52$</p>
      which is positive. The roots of the equation are, hence:
      <p style="cm">$x = ( neg(neg(8)) + root(52,2) ) / (2*3) = (8+
      <p style="cm">$x = ( neg(neg(8)) - root(52,2) ) / (2*3) = (8-
    </CMP>

    <CMP>
      The interested reader could try to copy the left hand side of
      into the function plotter to see if this matches the result.
    </CMP>
  </example>
<!-- http://localhost:8080/ActiveMath2/tutor/generateBook.cmd?name=dis
  </theory>
</omdoc>
```

Example for ' Theorem of Le Viète '

For example, we can compute the roots of the equation

$$3 \cdot x^2 - 8 \cdot x + 1 = 0.$$

The discriminant of that equation is:

$$\Delta = 8^2 - 4 \cdot 3 \cdot 1 \text{ hence } \Delta = 64 - 12 = 52$$

which is positive. The roots of the equation are, hence:

$$x = \frac{(-(-8)) + \sqrt[2]{52}}{2 \cdot 3} = \frac{8 + \sqrt[2]{52}}{6} \text{ or}$$

$$x = \frac{-(-8) - \sqrt[2]{52}}{2 \cdot 3} = \frac{8 - \sqrt[2]{52}}{6}.$$

The interested reader could try to copy the left hand side o the function plotter to see if this matches the result.

Figure 1. A typical screen layout of WYCIWYG: the files on the left, the content edition in the middle, and a preview on the right.

- the course generation can be run again to obtain a book taking advantage of the changed items

One should note that the perspectives above are quite diverse in nature but this list is only a sample of the aspects, under which an author might want to check the result of his authoring activity. The diversity of these aspects has justified what is currently felt as a limitation of jEditOQMath: the lack of an immediate preview of the result of the last authoring action following a reload. The author believes that the wealth of perspectives is close to how an author conceives the (intended) usages of the learning environment.

IV. Direct Manipulation for Authoring

This section reviews how direct manipulation, one of the principles of human computer interaction, which has been coined by B. Shneidermann in [3], applies to authoring of e-learning content. Direct manipulation refers to the immediacy of *effects* on the data in relation to the actions of the user. It has most often been used in a comparison between console interactions (such as a Unix shell) and visual paradigms (such as a desktop file system). The analysis of Frohlich [10] is more nuanced and summarises the lessons of experimental investigations following the direct manipulation proposal of [3]. Among these, one sees two claims [10, 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 quickly as possible. Similar practices can be found in user-interfaces for the TEX layout processor: the user's objective is the realization of the resulting view but his actions are carried on the source.

This approach can be opposed to the WYSIWYG approach, which requires the editing interface to be the same view as the *target interface*. This is doomed to fail when authoring for learning content on the web since *What You Get* cannot be captured in one view.

A more modern characterization of direct manipulation has been laid by the definition of instrumental interaction as in [11]: this approach describes the users as manipulating instruments, which are mediators to data objects: they operate commands on them and provide feedback to the user based on them. jEditOQMath can be analyzed by this method: the manipulation of the source is a manipulation of a readable and structured text that represents the content which the ActiveMath learning environment exploits. The source is a representation of the content that the instrument jEditOQMath offers for view and manipulation; another instrument of content manipulation is the author ActiveMath as described in section II. This instrumentalization 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 underlying OpenMath expression; we experienced the mapping of QMath to OpenMath objects to be easily explained but that QMath expressions cannot be taught alone, they need to be explained with the OpenMath understanding 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 instrument jEditOQMath limits its feedback to source and validation but that another instrument, the author ActiveMath, allows for a rich 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 [12], which considers the instruments as mediators that may evolve along the usage, the latter giving them their meanings: the author builds these meanings along his experiments.

We propose to name **WYCIWYG** the paradigm implemented in jEditOQMath: *What You Check Is What You Get*. The idea is that the editable objects of authoring are manipulated with the file and editor instruments with much of the feedback 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 an authoring tool view 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* [13].

V. Outlook

In this paper, the three authoring places have been introduced: the content collection files, the source editor, and the preview learning environment. They are depicted in figure 1. jEditOQMath is at the heart of these places: it is the editor, can open the files, and can request the server to process and reload so that the changed content can be viewed.

WYCIWYG is a significant paradigm to cope with complex, multi-dimensional web based learning systems, particular but not necessarily exclusive to Mathematics. Unlike conventional e-Paper, a web-based learning system exploits hidden features and dimensions (sometimes wildly) which cannot be made viewable all at once at a particular point in time. In order for the authoring process to be complete, it needs appropriate checks which the author performs by using the platform. We dare say that this paradigm is characteristic of the E-learning authoring activity, or, at least, of any authoring activity who has a similarly complex range of perspectives expected, all, to be used by some user.

A fundamental requirement of the jEditOQMath workflow to the **learning environment** is the support of **incremental changes to be previewed iteratively**. This requirement is, however, also important for the user interface of the learning environment for which ongoing work is being made. It is satisfied for several aspects of the learning platform such as book browsing, but some other features could honour it better. Among others the addition or removal of a collection, which needs server indexing and restart. Some exercise sequences or interactions with the course generator. Last but not least, little has been done towards modifying or experimenting with an arbitrary learner-model although many pedagogically relevant features of ActiveMath depend on its correctness. In many cases, an author has to redo many operations to achieve testing; research will show if **handles** to regain the perspectives may be available.

The preview and edit cycles we have described as routine operations following the WYCIWYG paradigm which is common in widespread authoring tools (e.g. DreamWeaver) but seems to be almost ignored in the authoring literature as examplified by most of the articles of the survey book [5].

A generalization of the WYCIWYG paradigm arises when different persons with different competencies are involved: for example an encoding person, a pedagogue 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 [14] is a start for the corporate learning world.

Planting roots of the authoring activity in the learning environment has a potential to transform consumers of the learning platform, from intensive users, through active sequencers, into active creators of learning experiences.

Acknowledgment

Research for this paper has been supported, among others, by the EU projects LeActiveMath (IST-507826) and Math-Bridge (ECP-2008-EDU-428046). The author wish to thank the many authors that have provided feedback to the making of this authoring tool, their trainers, Éric Andrès and Michael Dietrich, as well as AJG Bäumel and S. Sosnosvky for their comments to this paper.

References

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

[2] Michael Kohlhase. *OMDoc: An Open Markup Format for Mathematical Documents [version 1.2]*, volume 4180/2006 of *LNCS*. Springer Verlag Heidelberg, 2006.

[3] Ben Shneiderman. Direct manipulation: a step beyond programming languages. *IEEE Computer*, 16(8):57–69, 08.1983.

[4] C. Ullrich, P. Libbrecht, S. Winterstein, and M. 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 ICALT 2004*, pages 21–25, 2004.

[5] Tom Murray, Stephen Blessing, and Sharon Ainsworth. *Authoring Tools for Advanced Technology Learning Environment*. Kluwer Academic Publishers, Dordrecht, 2003.

[6] 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. [5].

[7] P. De Bra, N. Stash, D. Smits, C. Romero, and S. Ventura. Authoring and management tools for adaptive educational hypermedia systems: The AHA! case study. volume 62 of *Studies in Computational Intelligence*, pages 285–308. Springer-Verlag Heidelberg, 2007.

[8] 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.

[9] Alberto González Palomo. QMath: A human-oriented language and batch formatter for OMDoc. In Kohlhase [2], chapter 26.2. See http://www.mathweb.org/omdoc.

[10] Frohlich David. Direct manipulation and other lessons. In *Handbook of HCI: Second edition*, chapter 22, pages 463 – 488. Elsevier Science, Amsterdam, 1996.

[11] Michel Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-wimp user interfaces. In *CHI*, pages 446–453, 2000.

[12] 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.

[13] Mike Sharples. *How we write: writing as creative design*. Routledge, London, New York, 1999.

[14] Peter Loos, Volker Zimmermann, and Pavlina Chikova, editors. *Prozessorientiertes Authoring Management*. Logos Verlag, Berlin, 2008.