

Put me this space here !
(“but you want it re-usable don’t you ?”)

Paul Libbrecht

March 14, 2002

Abstract

We address here the difficult problem that authors of documents for the ACTIVE-MATH learning environment have to face: abstract encoding as defined in OMDoc lacks completely presentation facilities and for a good reason: re-usability. We try to define what kind of re-usability is meant here and under which circumstances it is possible to avoid the possibly poor appearance of a semantic encoding presented using an automatic engine.

The texts in italic-centered on top of sections indicate “goals” to be achieved in the underlying section. They are not meant for publishing

1 Introduction: presentation processes for OMDoc content

We introduce the presentation processes for OMDoc and their possible output targets.

The OMDoc content encoding is concerned with content only.¹ This means, currently, purely textual items (separated in well-defined items), mathematical formulas (encoded by their meanings) and metadata information for each of the items.

The presentation of OMDoc content to a readable form is obtained using a *presentation process*, such as the application of XSL stylesheet, which produces a presentation code (such as PDF or HTML) which becomes, then, suitable to display in appropriate viewers.

Let us discuss shortly the current advantages of a few presentation formats. Although PDF viewers are rather predictable in their behaviours they have almost no support for interactivity and other presentation mechanisms are much more suitable for a deployment, for example, on the web. For this purpose, the HTML presentation code is more appropriate. Display of HTML is, however, generally very poorly implemented in the major distributions.

¹quoting the OMDoc 1.1 manual, page xxx???

Alternative presentation encodings are available on the web with their corresponding viewer distributions: among others the multimedia oriented ones like MacroMedia Inc.'s FLASH, W3C's SVG or SMIL, and probably several others. They have each their advantage and seem to be more appropriate for a presentation of content where the positioning is important. We shall not cover these formats in more details if not mention that some attention will be paid to them within the ACTIVE MATH group in the future. A comparison between these formats and viewers, for the purpose of displaying interactively mathematical content will come in a later blue-note.

As one can understand, the presentation code is produced by a program, that is, it is not written by an author. Clearly, such a presentation process needs to be configured and enriched. Among others, the OMDoc encoding allows authors to define new mathematical symbols. Information on how to produce the presentation of these new symbols will have to be input at some point. OMDoc provides already a mild architecture for the enrichment of symbol presentations in XSL based presentation-process.

It is important to note, however, that this presentation process is the work of a developer, or, at least, of a stylesheet specialist. That person needs to know what is encoded in the OMDoc source and very precisely how the viewer and its language is working. He is then able to realize a presentation producer suitable for the given use, adding extra features and interactivity at will.

We shall, in this blue note, define the re-usability features, characterize the presentation mechanism in ACTIVE MATH, consider the possible ways to manually modify a presentation, then propose possible ways to integrate such modifications with a view on the lost re-usabilities.

2 Re-usability of mathematical content encoded in OMDoc

In this section we define the features of ActiveMath and OMDoc that pertain to re-usability. We describe which qualities of the abstract encoding are presented there. Here are a few things:

- *The multiple targets output (including browser bugs and export capabilities)*
 - *The reference abilities (eg for extending)*
- *The combination capabilities (eg with a course-generator or with a human combiner)*
- *The clean separation of the roles (content author, browser specialist and interactivity creator, server-developer)*

We start, in this section, with a description of the envisioned re-usability capabilities that the OMDoc content encoding offers, and present some of them at work in the ACTIVE MATH learning environment.

2.1 Multiple Output Formats

As we have seen, the presentation processes are multiple. For now, as running the ACTIVE MATH learning environment, HTML presentations, and a prototype PDF output using L^AT_EX are available. Under work are also an SVG and a FLASH presentation processes.

This capability to output to multiple viewers is precious when one considers the poor status of HTML compliance to standards in web-browsers: using slightly advanced features of HTML can break easily in some web-browser. The fixes needed to solve such a problem only need to be performed on the presentation process, avoiding this delicate task from an author that should be focused on producing content.

The capability to output in several formats is also interesting as an investment for the future: the possibly rich content being developed will not be lost because this or that viewer encoding is not usable anymore or because of a serious bug in a browser. The *no-prison re-usability* could be formulated as follows: if a content project ever had to leave OMDoc for some reasons, the team is not trapped in a proprietary encoding like the one used in MICROSOFT POWERPOINT or QUARKXPRESS: the format is well specified and writing converters to translate all the information encoded in another format is possible.

2.2 Separating the roles

When developing content in the old times, a book was written as a collaboration between the author and the typographer realizing in print characters what the author gave as manuscript. Before starting such an enterprise, the two persons needed to agree on a common language used on the manuscript. This language could evolve but had to stay understandable from both parties.

Presentation processes reproduce this contract between an author and presentation process developer: the *Document Type Definition* (a DTD), and the content encoding specifications, define the information available to present and the rules a content should be written with. Using this contract, possibly helped by validation tools, the author knows the amount of information he can and should encode. Using this specification and the DTD, the presentation-process developer knows which information he can present and take advantage of it to provide to the user of the presentation viewer, the most appropriate view. Results are obtained by combining content and presentation process. Both of these relatively independent parts can then evolve separately towards the same goal.

In other publishing practices, the separation is often much less clear. A rather sad example is the editing of HTML web-sites or multimedia presentations as currently made in the industry: the content should be fixed in advance, the HTML developer or artist produces the layout, work may be then followed by the server-side developer whose work to encode this in a functioning program. Evolution in such a sequence of works is hard and may often require to pass through the whole process again.

2.3 Personalizing the Presentation

As we have seen the presentation is obtained through the application of a program. Adding parameters to this program allows the presentation to be customized by the user. For example, in the current ACTIVE MATH learning environment, two “appearance themes” can be chosen from which will affect the HTML presentation of items. Moreover, ACTIVE MATH allows multiple-language OMDocs to be presented: the language used in the page presented is decided dynamically (that is, on presentation time) as a balance between the available source languages and the preferred language of the user. Many other customization can

happen: adapting the presentation to the screen-size or page-size and to the preferred font-size is an essential flexibility that will have to be built. Also, it may happen one day, that ACTIVE MATH allows single users to overwrite the rendering information of a particular symbol so that it is displayed according to the taste of the user. We shall call this re-usability, the *personalization-reusability*.

2.4 Combining materials

We now come to features which are closer to actual re-use of content. As we are concerned with learning material, let us state a common practice: teachers tend to re-use at least parts of the courses they already made in other courses they give. In an electronic media publication, this would involve taking an existing version of the course, performing a few additions, removals and/or corrections. It is rare however, that a teacher takes courses given in several different years and combine them into a new one: the merging work is too large and the redundancy can become unwildly to manage.

•check en-
glish

One important facet of OMDoc encoding is the ability to declare unique identifiers for fragments of content. Of particular interests are fragments that can have a logical or pedagogical meaning; in the classical mathematical education, such fragments often fall in categories such as *definition*, *theorem*, *example*, *proof*, *motivation*, etc. We shall call these fragments *items*.

If a content is divided in such well-defined items, re-usability following the quoted teachers' practice reaches a higher level of efficiency: preparing a course with existing material would essentially mean to decide on which item to use at which course, modifying a few of them in a conservative way, and adding a small amount of new items. We shall call this feature, the *combination re-usability*.

As an addition, OMDoc content can be enriched with metadata. ACTIVE MATH has developed the metadata scheme of this language to allow an intelligent content choosing process called the *course generator*. Using metadatas, an up-to-date model of the user's knowledge, and configurable pedagogical rules, a book can be prepared individually for each student containing only the material that is appropriate for a given set of goals and for the current user's knowledge.

The combination re-usability can actually go beyond the practice of a single teacher: OMDoc content is encoded in a way that makes it particularly easy to exchange documents: they are self-contained and don't rely on an external content (such as a L^AT_EX style-file). Hence, a given teacher is able to use the content of another author in his course, either by simply preparing a book that contains chosen items from a few sources, or by combining selected items of existing contents with his own. We shall call this re-usability the *editing re-usability* and shall name the role of the person performing such a selection the *editor* role.

The editor task goes slightly further than collecting pointers to items in a table-of-content, it also involves merging the presentation information of symbols (see §1) into the presentation engines that will be used in the foreseen installation. As symbols are declared with a unique semantic in OMDoc, such a merging is a mild task. More work is required, however, if symbol semantic has to be aligned, that is, if symbols in one collection has to be the same as symbols in another collection. Tools to perform such merging will be developed within

the ACTIVEMATH group. For team-developed content, however, the unique referencing mechanism provides the complete architecture to allow authors to refer to others' contents: without the content needing to be ready, an author can right-away refer to another's symbol or item and use it as if it were part of his content.

3 The ACTIVEMATH presentation mechanisms

Describe first the characters and goals of the presentation mechanism Continue with a kind of prototypical architecture of a proposed new one including the grabbing, pre-processing, stylesheet, and post-processing steps.

We then focus on the ACTIVEMATH presentation process for output of HTML and L^AT_EX.

ACTIVEMATH is a learning environment presenting mathematical content in a web browser, offers interactive exercises online, is equipped with a user-model that tracks reading and exercise achievements, and presents content either adapted to each user's knowledge or as encoded in pre-made table-of-contents.

The ACTIVEMATH presentation process is triggered by an HTTP request of a browser which would like to display a page of a book whose table-of-contents is known, or a page of the ACTIVEMATH *dictionary*. The following steps are performed:

TODO: include a graphics of these architectures

TODO: make these steps more detailed

- *[fetching]* collect content from the database of OMDoc items and gather this into one OMDoc document (this step only depends on the table-of-content)
- *[pre-processing]* go through the document so as to insert information in the OMDoc that are system dependent (this should only be parametrized by the current installation, preparing, for example, links or resource descriptions to interactive exercise sessions)
- *[transformation]* Perform a first transformation (like the application of an XSL stylesheet to the document). The number of parameters needed in this phase is critical to the flexibility of the system, as we shall see.
- *[personalization]* Apply personalization for the given user so that: language is chosen, links are keep tracks of the users (e.g. for user-modelling reports) and so that appearance personalization is performed.
- *[packaging]* Apply further processing to convert the content presentation into a format that can be displayed by the client

Currently, ACTIVEMATH has two presentation processes which produce, for the first, HTML 4 presentations using Unicode symbols and cascading-stylesheets for mathematical symbols, and, for the second, a PDF output meant for printing.

The last step, the packaging step, is reduced to zero in HTML presentation process and involves the application of the `pdflatex` program for the PDF output.

In the two presentation processes, the two middle steps, XSL and personalization, are currently performed at the same time but, as we shall see later, it may be wishable to allow the personalization to be a third step.

The quality achieved so far is typical of an automatic presentation: line-breaks, for example, are extremely poor in HTML which allows itself to cut in any location of formulas, line-height is poorly managed as well (an simple exponent can make the line become suddenly higher), page-breaks in the PDF output are only weakly controlled, images in the PDF output are considered as *floating objects* hence can sometimes be relocated at fancy places.

In general however, the results achieved thanks to the (pdf) \LaTeX program are incomparable to the HTML output and, for this reason, future developments of presentation processes within the ACTIVE MATH group will mostly focus on presenting the result of a \LaTeX output on-screen.

Adjusting the presentation process to accomodate a new interactivity feature is mostly done by adapting in the transformation or presentation steps, and possibly the packaging step. The same applies to adjustments needed to accomodate bugs appearing in some target-viewers. The fact that such adjustments have to be done only on the presentation process is a major advantage for development: these adjustments can be performed at the same time or later than the content development.

4 Manually modifying the results of the process

We discuss the various places in the automatic presentation process where a (picky) author could modify manually content so as to influence the resulting presentation. The lost re-usability features are highlighted.

We now try to address where, in these presentation processes, a teacher or author that insists on the fine-grained quality of layout of presented material can act to adapt the presentation manually. (?? TODO: find him a name ??)

The first candid action such a person could do would be to use the presentation engine to produce a book as a finished set of HTML pages or a \LaTeX sources and edit them manually, possibly producing a well-controlled PDF output thereafter. For them to be available on the web, he could only publish them through a classical web-serving engine.

Such modifications clearly allow any modifications to be performed. Adjustments for a floating image to appear at the right place can be made, page breaks can be taken care of with precision, breaking the lines of a too long formula can be done as most appropriate as possible.

Personalization (see §2.3) is completely lost as the content is now static (for example, the user's preferred screen or paper width cannot be honoured). Moreover the time spent to perform these modifications should be kept to a minimum as nothing of these changes can be re-used in any later deployment except for simple copy-and-paste in the old way quoted in §2.4. That is to say the combination re-usability is also lost and, if a bug in the viewers appear (for example a font-encoding problem), that person will need to modify each produced outputs.

A better approach would be to modify the presentation of individual items. Doing so would only solve a fraction of the problems (at least page-breaks would remain weakly controllable). But it may allow, if the architecture is prepared for it, to keep the modifications so that they could be re-used when combined to other items.

It is to be noticed that such modifications will be performed with a great number of fixed information that could be, otherwise, customized for the user or for a given installation. An example goal could be to solving manually the formula line-breaking problem. Such modification contribute to a greater quality of the presentation but only makes sense with a fixed size of the screen or paper, a fixed set of symbol-presentation, and a fixed font.

The editing and combination re-usability (see §2.4 and §2.4) would, however, be preserved. The usage of a manually adapted items in a presentation can however run the risk of a lack of graphical homogeneity, a lack that is considered a mistake in professional publishing.

5 Suggested enhancements to honour the promise

We propose changes in the ACTIVEMATH presentation architecture to allow manual changes made by an author and their appropriate storage.

As of this writing, the ACTIVEMATH presentation architecture is in a state requiring a deep re-engineering so as to admit the easy addition of new presentation processes. The architecture depicted in §3 is (?? probably ??) the one that will be implemented.

One of the possible enhancements for speed that this renewed architecture could provide is the ability to cache the results of each steps so as to save resources. For example, the result of the application of the XSL stylesheet for each items could be cached and re-used by all other requests that would serve this item. For this to happen, the steps until the transformation step have to be made without any personal parameters, which involves a great abstraction in the process.

Supposing such a cache can be achieved, the same substitution mechanism as a cache can be applied with manually manually edited content: when applying the process, the fetching step may detect that a manually edited item's presentation is available and request that this one be included after the transformation process.

We recommend that such a practice would only be adopted with a very small amount of personalization. This is the case, for example, in the current HTML output of a single-lingual OMDoc as personalization in this process is restricted to URLs and headers rewriting.

6 Conclusions

Mention current status and future works quickly. Mention the works made towards a more readable L^AT_EX output on the screen

This blue-note has attempted to raise the delicate question of a manually controlled presentation in an environment where most of the presentation is automatized. It has proved that the re-usability capabilities provided by an automatic presentation process are dangerously threatened by attempts to modify manually the output.

Currently in `ACTIVEMATH`, only the `HTML` presentation process is mature and full-featured. The presentation processes that are based on `LATEX` will most probably be worked extensively as mathematical formulas it produces have an outstanding quality compared to `HTML` rendering (and probably to most `MATHML` rendering as well).

Presentation on the screen of `LATEX` documents, at least using the standard Computer Modern Roman set of fonts is, however, being recognized to be inappropriate because of its too large xxx (?? empattement ??). Attempts to obtain more readable presentation have started (as for example the `MMISS-LATEX` style for slides) and will continue.

However manual modification of a content may also loose the re-usability of an evolving viewer language (see §2.1): such manual modifications will be made useless when a better font is chosen to present the content.

7 Acknowledgements

This blue-note emerged from a fruitful but too small discussion with Holger Schlingloff, TZI and Universität Bremen. We thank him here.