

Towards a Mathematical Clipboard on the Web

Paul Libbrecht

This article is submitted to the third Mathematical Knowledge Management Conference, <http://mizar.uwb.edu.pl/MKM2004/>.

DFKI and Universität des Saarlandes, Saarbrücken, Germany

Abstract. The copy-and-paste paradigm is one of the most natural ways for users to exchange information between different places in a user-interface. Naturally, users often expect to copy anything viewable including mathematical formulæ. In the development of a web-environment where mathematical content is shown, we have thus tried to provide the capability to copy-and-paste mathematical formulæ from presented content to input places in interactive exercises.

This article describes the limited implementation of copy-and-paste performed in the ActiveMath learning environment, as well as the issues and wishes met thus far. In particular, it explores approaches close to web-services whereby a source of mathematical content on the web like ActiveMath could be used to offer its information to other mathematical systems.

[...] Having found the relevant information, you would have to copy this into your clipboard (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 [11]*

Introduction

To input mathematical formulæ is a known difficult task: experts use character-line-based input-syntax to do most of the job, while most beginners tend to use buttons layered into palettes with a visual display of the formulæ.

Copy-and-paste of mathematical formulæ can come as handy help to help input. The needs for it has been quoted in many places in the literature:

In their efforts to provide user-friendly exercises in their webMathematica-exercises, Albano, D'Auria, Pagano and Viglione ([1]) pinpoint important facilities needed in order for the users to spend less time on inputting their mathematical objects and more time on thinking on them. The facility to copy-and-paste mathematical formulæ is one of the wished features.

In most *advertising* papers around MATHML, it is claimed that copy-and-paste from content on the web to mathematical or typesetting systems is or will be possible. See for example [12], [19], [21], or [5].

In their vision of the semantic web, [11], T. Berners-Lee, E. Miller, and J. Hendler state why the semantic web should allow users to intuitively copy-and-paste *semantic data*. Mathematical objects presented in mathematical formulæ are just an example of semantic data. To the knowledge of the author, little seems available for this purpose.

The ActiveMath learning environment is a web-based adaptive intelligent learning environment presenting mathematical documents to learners along with interactive exercises. The semantically encoded content in ActiveMath is a key ingredient to offer copy-and-paste of mathematical formulæ between the presented content and mathematical systems. ActiveMath includes front-ends to computer-algebra-systems used within exercise scenarios; this is where learners can now paste content they have copied.

This article intends to provide a description of the copy-and-paste prototype provided in ActiveMath and compare it to other existing copy-and-paste facilities for mathematical content. It continues with an exploration of possible futures with respect to copy-and-paste of mathematical formulæ, arguing that the problem is generalized to the semantic web.

1 Copy-and-Paste Principles

The copy-and-paste paradigm is a form of data-transfer between applications. The user first selects a piece of information, she then invokes the *copy* or *cut* action which is considered to transfer the content into a single place called the *clipboard*. Later, she can invoke the *paste* action which inserts the content *at the insertion point*.

When the paste-action has occurred, the receiving application queries the clipboard in order to insert what the user expects to be pasted. For textual or bit-map content, this is mostly problem-free as it involves little conversion. For other type of content, however, a negotiation has to happen in order to decide what to paste. This negotiation generally uses the notion of MIME-types: the receiving application inspects the MIME-types available, then chooses the one it expects to fit best and requests the content in this form.

The *drag-and-drop* paradigm uses a similar data-transfer. It is based, however, on the metaphor of a physical move from one place to the other in the user-interface: the user *drags-out* an object and moves it toward its target. The mouse-pointer then provides her feedback, indicating the possibility to drop the indicated object to the *drop-target*. In order to provide this feedback, again, only MIME-types can be inspected.

MIME-types are a typing mechanism that was built to give types to e-mail parts. It is documented in several RFCs, among others the RFC-2046 [8]). MIME-types, among others, have attempted to distinguish text, image, and *application-level* fragments and have striven to provide a list of all existing data-types. The advent of XML has brought difficulties to this: an XML document is a text-document, it is however not intended to be read by users and, hence, was

classified in the *application* side.¹ In general, MIME-types offer a coarse-grained classification of content and this coarseness will be more and more apparent as the semantic-web technologies will allow several types of content to be merged and combined.

2 Copy-and-Paste of Mathematical formulæ Existing Solutions

We provide an overview on existing approaches of copy-and-paste between mathematical systems. As the amount of mathematical systems is too vast for a complete survey, we focused the systems which seem widely used.

2.1 Between Commercial Systems Pairs

In the input-answer paradigm taken by most mathematical systems, an input syntax visible to the user is central. The data exchange between various places of the user-interface are thus best understood as exchange of the textual content.

This closed world works very well within contemporary computer algebra systems like Maple, Mupad or Mathematica. It also works well from places in their user-interface where a formula is presented typeset. However, transfer between them does not work uniformly. Transfer to and from commercial typesetting systems such as Scientific Workplace, TeXMacS, or MathType may work. To the knowledge of the author, in most cases the exchange works only by a common agreement between the implementers.

2.2 Integrated User-Interfaces

In [13], Kajler and Soiffler provide an extensive survey of the existing user-interfaces for computer-algebra systems available until 1995. It covers their selection mechanism, and their ability to perform copy-and-paste. In this paper, exchange between systems is not treated and we dare expect it was as limited if not more than our description of mainstream systems. Prototype systems integrating several computer-algebra-systems with exchanges between them are, however, described. The conversion processes are explicitly quoted as a delicate part.

2.3 On the Web

The world-wide-web is a very rich source of mathematical content about almost any topic. Some collections have a very restricted content-encoding, for example MathWorld [20] for which everything is encoded in Mathematica and exported. Some control it much less (for example the The Geometry Junkyard of David Eppstein [6]).

¹ See RFC 3023, [18]

Several users would naturally expect the mathematical objects found in these collections to be re-used with copy-and-paste in many applications, be them computational programs, typesetting programs, or other interactive programs. Most of the time, the result of such a copy is, however, a plain text string which has rarely a suitable form for any kind of further usage and may even, with some symbols, lead to subtle errors due to incompatible interpretations of symbols.

The MATHML [3] and OPENMATH [2] standards provide an encoding which should allow applications to exchange content easily. Requirements of MATHML actually state this feature [19].

MATHML is made of presentation-markup and content-markup with abilities to synchronize the two markup-trees. It succeeded with cross-platform implementations in short time; its success mostly comes, however, from the presentation capabilities. In known platforms displaying MATHML, copy-and-paste was not very developed thus far:

- being able to select a *sub-term* of a presented formula is only possible in Amaya or WebEQ. The latter allows content-irrelevant selections (i.e. *presentation only* selection like $1+$ is $1 + 2$).
- being able to copy the MATHML snippet is offered for complete MATHML expressions only (that is, for complete formulæ), and needs the user to view the source in Mozilla, except for WebEQ which will copy sub-trees even with the matching content.

The result of the copy-action is *put* in the clipboard as simple text. Recipients of the clipboard can analyse this clipboard, XML-parsing can then be attempted. The fact that the plain-text MIME-type is used makes the paste experience relatively random. Either users face the fact that, sometimes, the MATHML pasted is shown as part of the input (which commonly spans more than 30 lines), or they use a special paste command to paste a MATHML snippet.

2.4 Using Content-MATHML

The interpretation of the paste or import of MATHML should (and sometimes does) use the content sub-tree. Often, however, depending on the system used, the presentation sub-tree is used along with much heuristics and a bit of artificial intelligence, to produce the content using the internal syntax. The heuristics thus bundled is bound to the K-12 level aimed at by MATHML.

An essential fact of MATHML is to allow sub-terms to be correctly selected and correctly copied: MATHML has a way to *synchronize* XML-trees in the presentation and in the content (or in any other XML-based semantic part, like in OPENMATH). This allows browsers to match the presentation markup with the content markup and vice-versa.

Receiving MATHML-content for a mathematical system will be always limited by differences in the definition of functions: the MATHML-standard defines a limited set of mathematical symbol. If a copy-source has a different definition of a function, it is free to use, at the content-level, a symbol with a private definition. MATHML provides the `csymbol` element for this purpose.

This would guarantee the consistency of the content being copied, hence the mathematical correctness of further processing. However, it would require other systems to be able to understand these symbols. Only a limited number of systems have ways to specify how other systems' `csymbols` should be interpreted. One clearly expects standardized ways to provide such definitions. Such definitions have been among the purposes of standards like OPENMATH content-dictionaries ([7]) or OMDOC ([14]). We report below about our experience with these standards.

3 Copy-and-paste in ActiveMath

ActiveMath is an intelligent web-based learning environment. It presents mathematical content to the learner in a user-friendly fashion and combines this presentation with interactive exercises which may integrate mathematical systems.

Several reasons, ranging from re-usability and clean separation of concerns between the authoring and developers' tasks (see [17]), have lead us to choose OMDOC as the underlying knowledge representation, [14]. It structures the units of content in well-defined mathematical items and encodes mathematical objects in OPENMATH. The latter is a standardized encoding of mathematical objects, supported by dictionary-like declaration of mathematical symbols in the form of content-dictionaries (see [7]). OMDOC incorporates OPENMATH's content-dictionaries and extends its way to define new symbols in a more general framework where definitions and theorems can be provided as part of documents.

The OPENMATH collection of symbols is somewhat similar to the set of symbols one finds in contemporary mainstream computer-algebra-systems. This similarity is made stronger by the fact that OPENMATH supports *phrasebooks* that can convert OPENMATH to and from several computer-algebra-systems languages.²

3.1 A word about ActiveMath's presentation system

Presentation of OMDOC content in a user-friendly fashion thus needs a *presentation process* where the mathematical objects become visually presented according to *presentation rules*. The presentation process of ActiveMath is explained in [10] where one can read how an XSLT [4] stylesheet converts the mathematical symbols to presentation markup and how this conversion is built into a process producing a user-friendly presentation of mathematical content on the web (see also [17]).

When authoring mathematical documents, the authors often require new mathematical symbols. OMDOC extends the way symbols are declared in OPENMATH, allowing this declaration as part of the documents, offering space for one or several definitions along with supporting mathematical developments such as

² phrasebooks are defined in the OPENMATH specification [2], several implementations exist.

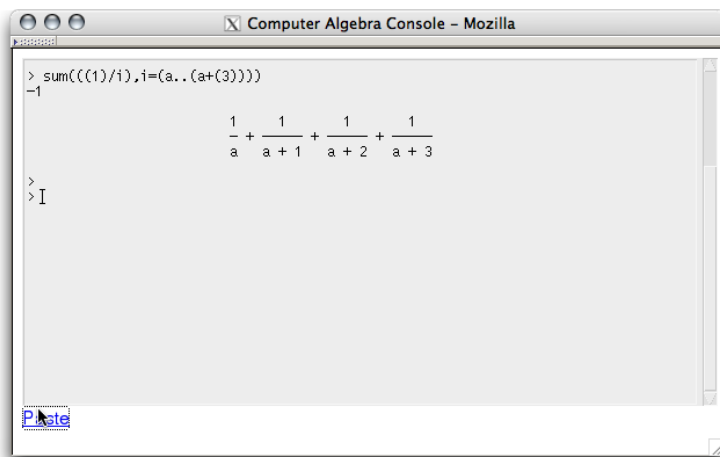
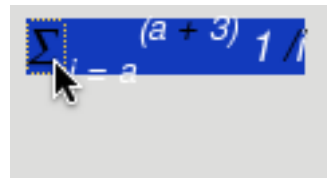
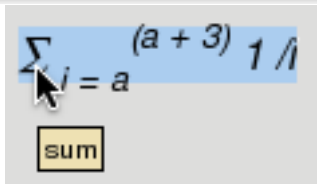
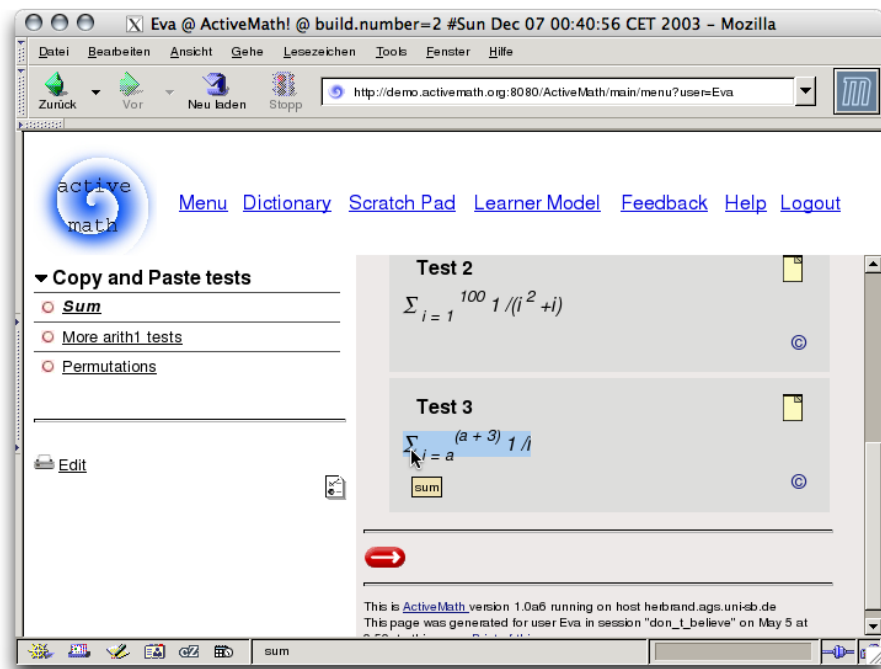


Fig. 1. Sequence of steps to copy a formulæ from the example page and paste it into the MuPad console: overall book, sub-term highlighting, formula selection, paste-action followed by a press of the return-key.

theorems to justify this definition. The declaration of a new symbol, when the symbol is used as part of a formulæ, must be made along with an extension to the presentation rules so as to provide a notation.

Using this presentation system, we thus have an extensible presentation engine which offers to learners a user-friendly presentation of mathematical objects. It supports tool-tips indicating symbol-names, clickability to navigate to definitions and usages of a symbol, and sub-term selection. The sub-term selection mechanism offered to the learner is a direct spill-off from the content-encoded source: it highlights the smallest subterm (that is, the content-sub-tree) containing the area under the mouse. A small *drag* then chooses this sub-term as selection. The content is considered to be copied. This sequence is pictured in Fig. 1.

3.2 Interactive Exercises in ActiveMath

ActiveMath intelligent behaviour is based on a user-model which stores and updates estimated user's mastery for each concept of the content for each user. The updating happens with respect to the learner's performance in interactive exercises.

ActiveMath provides classical multiple-choice-questions exercises. As the student's input is restricted to selecting between predefined alternatives, we shall not discuss them any further.

ActiveMath also provides interactive exercises where the learner is expected to input formulæ which will be evaluated by a mathematical system. They have been described in [15]. In order to input these formulæ, the learner currently has to use an input syntax. Mastering the input syntax requires from her an extra effort in a learning situation where the working memory is already quite loaded.

Therefore, copy-and-paste comes as a very precious help: the learner can browse the learning-content and find there extracts of formulæ she considers similar to what she would like to input. Using copy-and-paste, she obtains the input syntax through the paste command and can modify it to suit her will.

3.3 Technical Description of Copy-and-paste in ActiveMath

In the following, we will describe the steps involved in copying a piece of presented content and pasting it in computer-algebra-system console:

1. At presentation time, each node of the content-tree is converted to presentation with a track of its coordinate (which is an identifier followed by a child-order sequence)
2. Once the markup is displayed in the browser, the nodes of the presentation tree can be interpreted for their correspondence to the content-tree node and the smallest sub-term under the mouse can be highlighted.
3. A small *drag* of the mouse sets the selection

4. In the MUPAD-console (the exercise type for which copy-and-paste works thus far), the learner can then press a *paste* link which can collect the content-node and requests the applet to paste the equivalent of the content of the indicated node.
5. The applet then communicates to the server which extracts the OPENMATH node at the given coordinate and applies OPENMATH phrasebooks to convert it to the appropriate input-syntax
6. The value is returned to the applet which can insert it at the insertion point in the MUPAD-console.

This can be tried in the *old* presentation system in the ActiveMath official demo: <http://demo.activemath.org:8080/ActiveMath/> where a book has been prepared for the purposes of testing copy-and-paste.³

3.4 Drawbacks of the Copy-and-Paste in ActiveMath

Using this sequence of steps, we reach a first prototype of copy-and-paste with limited generality. It is far from being intuitive for the learners as it is not using the user-clipboard nor its mouse or keyboard gestures. This is due to the fact that the system-clipboard is not accessible to code running in an untrusted environment such as java applets or javascript code.

A drag-and-drop paradigm might have been more intuitive for learners but other technical problems have arisen: the only data that can be dragged out of a browser window is a URL (as obtained when dragging a link denoted in an anchor element) and dropping such a URL in a browser window... opens this URL instead of offering the document's javascript or applet component the chance to receive the drop.

The current implementation is also fragile in the sense of indicating to the user the process of converting the information for the paste to happen: locking the console while the conversion happens is not done and would be very long, moreover failures are only signalled by a beep.⁴ A revised exercise-architecture has been started in ActiveMath, see [9], it shall be more flexible and should offer the necessary client components.

4 Extensible Conversion of Semantic Mathematical Content

Even if the usage of the copy-and-paste mouse gestures to be used in ActiveMath, as we have described in the previous sections, was taught to learners, the paste would still fail if newer symbols were used. The reason for these failures is, on

³ Copy-and-paste currently has been tested successfully only on Linux, Windows, and Solaris using Mozilla 1.4 and later, and a Java plugin version 1.3 or higher. On Mac OSX, positive results were obtained irregularly.

⁴ The development of this console has been frozen recently. This is due to the fact that the component was based, erroneously, on a java AWT component

the one side, the limited symbol-set covered by phrasebooks and, on the other side, the desire of authors to permanently create new mathematical symbols.

There exists several phrasebooks, each is very specific to the underlying mathematical system (and may even require it running). The set of mathematical symbols supported by a phrasebook is inherently limited by the set of mathematical objects one can represent in the language of the mathematical system (for example, the GAP computer-algebra-system, specialized to computations for groups, will not understand the symbol of integral or of gamma-function).

The desire to create new symbols in mathematical documents is omnipresent: among others mathematicians tend to adapt their notations to reflect new concepts they introduce, including specialization of existing symbols in meaning or in appearance. In the ActiveMath authoring workflow, declaring a new symbol involves declaring an OMDOC element representing this symbol followed by an extension of the input syntax and of the presentation process. The input-syntax is based on QMath⁵ which accepts extensions almost at any point in the processed document. The presentation rules to convert the OMDOC fragments to the presentation language in ActiveMath need to be enriched to be able to process the OMS elements which denote usage of the symbol in a formula. Simple support to produce the necessary XSLT templates is provided in OMDOC in the form of *presentation tags* which can be converted to XSLT templates. These two extensibility facilities are appropriate to author content: the declaration of a new symbol is made along with the extension of the input-syntax and the one of the presentation system.

Conversion processes to and from semantic mathematics are numerous, be them phrasebooks or others. Extensibility is handled differently in each product, rarely is this as much declarative as a piece of XML content. Among others, the RIACA phrasebooks⁶ use simple java-classloading which can be slow and fragile.

Ideally, such an *notation extension* should allow the author declaring a new symbol to specify ways to present it (that is to convert it from OPENMATH) and ways to input it in all input forms. Input forms should include, when applicable, the input-methods for the authoring tool and the input-method for interactive exercises. Of course, for teachers responsible to deploy a mathematical course, it is hoped that such declarations are consistent and intuitive, hence the desire to centralize them around the symbol declaration.

5 Transferring Semantic Content on the Web

The copy-and-paste mechanism in ActiveMath has another weakness: it cannot be applied in order to exchange mathematical objects to or from external programs running on the learner's machine. The security restriction of a web-application is the first reason of this problem. There, is however, a deeper reason: if ActiveMath does not know where the object shall be pasted (or dropped), it

⁵ See <http://www.matracas.org/>

⁶ See <http://www.riaca.win.tue.nl/>.

will not know how to convert it, hence will not know what to provide, for example, for the `text/plain` MIME-type alternative (should this be Maple, $\text{T}_{\text{E}}\text{X}$, MATHML or Mathematica syntax?). Furthermore, supposing the MIME-types would be much finer and be specified for each mathematical system language, ActiveMath would need, at time of clipboard export, to convert the object into many languages which is highly inefficient and very resource-intensive.

Such a conversion problem is not specific to ActiveMath and is not even specific to mathematics, it is general to the semantic-web efforts. Currently, the semantic-web practice would tend to promote the use of sharable ontologies on which applications agree. While such *dominating* or *unifying* ontologies are still being discussed about, it is probable that conversion processes will remain. However, there seems to be little made about defining the extensibility for conversion processes.

To provide a parallel with the classical examples of travel-planners in the semantic-web vision articles, we describe what would be the semantic information for a physical location on the earth. A *location* is a fairly general term and its semantic encoding can have many different representations: for some applications, a longitude and latitude pair is the most appropriate formulation (this is the case for meteorological services for example), for other applications (like bus-planners or packet delivery service), a postal or residential address is much more appropriate, finally, for some purposes, travelling directions may be most efficient. For an address-book application, or a congress web-site, it is not clear which encoding should be chosen to advertise to other applications. Moreover, a conversion between one and the other is actually an expensive computation requiring important knowledge-bases. Hence the need for such an address-book application to be aware of the recipient of the paste action to invoke the necessary conversion (probably as web-services) when pasting.

5.1 Copy by Reference or Copy by Duplication

When using copy-and-paste, it is generally accepted that the content is duplicated before being inserted in the receiving applications. When performing drag-and-drop, however, the source information can be either copied or given as reference; the decision between the two is generally done by a system-specific gesture such as a key-press. We call *copy-by-reference* a copy command where only a reference to the content is put into the clipboard. An HTTP URL is a natural candidate to represent such an reference.

Copy-by-reference is a practical solution for our current problems: encoding the content of the selection as an HTTP URL would provide receiving applications with the ability to request the content to be pasted by querying this HTTP URL. The sending side has the ability to use the receiver's identity and supported MIME-types to provide the conversion which best suits.

For example, in the case of ActiveMath, a computer-algebra-system like Maple would be able to query these URLs by indicating it knows Maple-syntax, it knows MATHML-content, and at worst knows MATHML-presentation. The ActiveMath server would then be able to see whether it has a Maple phrasebook

at hand and attempt the conversion of the selection to Maple syntax (which may fail), then to MATHML-content (which may fail in Maple because of unknown `csymbol` elements), then to MATHML-presentation which may succeed but may also be wrong as this conversion is based on heuristics.

In his presentation about interfacing with GAP [16], Steve Linton explained well why transferring an explicit representation of a group outside of GAP was a generally bad idea: it can be summarized in that the constructors needed to support all groups are very numerous (hence content-dictionaries would be huge), and that little can be said about a group given some constructors, unless some expensive computation is actually done. This seconds largely that copying-by-reference would be an appropriate way to transfer the presentation of a mathematical object declaring a group.

Supposing that the presentation of this object is actually generated by GAP (and converted to presentation as a normal part of an interactive experience), the fact that a reference is pasted into a receiving application (for example another GAP instance or a typesetting system) actually opens the door to web-service type of exchanges: the URL could be the URL of a web-service around this GAP instance which could then be queried for facts about this group, be requested extra-computations like the extraction of a character table of this group, or, in the case of real functions, the request to provide a graph, etc.

Conclusion

We have described how much richer the ActiveMath copy-and-paste of mathematical formulæ can go compared to other technologies on the web.

The prototype implementation of copy-and-paste we have reached in ActiveMath has highlighted two issues which seem to characterize the copy-and-paste mechanism for any mathematical content on the web and probably any presentation of semantic information:

- The declaration of a new symbol, be it in an OPENMATH content dictionary, in an OMDOC symbol element, or as a simply new URL (as in MATHML-content) is best made along with extension of the conversion processes to and from the semantic mathematical language. Such an extension can be seen as the *definition of a notation* and should be in a central location in the authoring phase.
- classical copy-and-paste tends to proceed through a *duplication of content* where the data to be copied is only documented through its MIME-type. We have shown how impractical this situation is for mathematical objects which can have many different encodings. We have described how an URL exchange followed by HTTP-requests would provide practical solutions and have shown that a well-informed content negotiation is imperative when activating the copy-and-paste data-exchange. This negotiation seems to be best formalizable as a web-service and could then offer new facilities.

Thanks

The copy-and-paste mechanism is the result of the work of several persons in the ActiveMath team, directed by Erica Melis. In the group, particular thanks for copy-and-paste go to Vladimir Brezhnev which has first performed the implementation, the students who failed using the phrasebooks for this purpose, and to Carsten Ullrich and George Gogvadze having provided the XSLT transformation to convert the OMDOC content to a user-friendly HTML presentation.

References

1. G. Albano, B. D'Auria, A. Pagano, and S. Viglione. How to achieve user-friendly interactivity in mathematical exercises on the web. Talk presented in the Mathematics on the Semantic Web Workshop (<http://www.openmath.org/meetings/eindhoven2003/index.html>), Eindhoven, May 12-14 2003.
2. O. Caprotti and A. M. Cohen. Draft of the open math standard. Open Math Consortium, <http://www.nag.co.uk/projects/OpenMath/omstd/>, 1998.
3. D. Carlisle, P. Ion, R. Miner, and N. Poppelier. Mathematical markup language, version 2.0, 2001. <http://www.w3.org/TR/MathML2/>.
4. James Clark. XSL transformation, Nov 1999. World Wide Web Consortium, see <http://www.w3.org/Style/XSL/>.
5. B. Cogan. Software reviews: webmathematica. Scientific Computing World, See <http://www.scientific-computing.com/review5.html>.
6. D. Eppstein. The geometry junkyard, May 2004. <http://www.ics.uci.edu/~eppstein/junkyard/>.
7. J. Davenport et al. Openmath core content dictionaries, April 2004. See <http://www.openmath.org/>.
8. N. Freed and N. Borenstein. Multipurpose internet mail extensions (mime) part two: Media types, November 1996. <ftp://ftp.rfc-editor.org/in-notes/rfc2046.txt>.
9. G. Gogvadze, E. Melis, C. Ullrich, and P. Cairns. Problems and solutions for markup for mathematical examples and exercises. In A. Asperti, B. Buchberger, and J. H. Davenport, editors, *Proceedings of Second International Conference on Mathematical Knowledge Management, MKM03*, LNCS 2594, pages 80–93. Springer-Verlag, 2003.
10. A. Gonzalez-Palomo, P. Libbrecht, and C. Ullrich. A presentation architecture for individualized content. In Paul de Bra, editor, *Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems, AH2003*, pages 87–98, 2003. See also about <http://wwwis.win.tue.nl/ah2003/>.
11. J. Hendler, T. Berners-Lee, and E. 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>.
12. Wolfram Research Inc. Mathml central: Mathml history. See <http://www.mathmlcentral.com/history.html>.
13. N. Kajler and N. Soiffer. A survey on user interfaces for computer algebra systems. *Journal of Symbolic Computation*, 25(2):127–160, 1998. See <http://www.enscm.fr/~kajler/bibliography.html>.

14. M. Kohlhase. OMDOC: Towards an OPENMATH representation of mathematical documents. Seki Report SR-00-02, Fachbereich Informatik, Universität des Saarlandes, 2000. See also <http://www.mathweb.org/omdoc>.
15. P. Libbrecht, A. Frischauf, E. Melis, M. Pollet, and C. Ullrich. Integration of mathematical systems into the activemath learning environment. In Paul Wang, Norbert Kajler, and Angel Diaz, editors, *ISSAC-2001 Workshop on Internet Accessible Mathematical Computation*, 2001. <http://icm.mcs.kent.edu/research/iamc01proceedings.html>.
16. S. Linton. Interfacing with gap. Talk presented in the Mathematics on the Semantic Web Workshop (<http://www.openmath.org/meetings/eindhoven2003/index.html>), Eindhoven, May 12-14 2003.
17. E. Melis, J. Büdenbender, E. Andrès, A. Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. Knowledge representation and management in ACTIVE-MATH. *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>.
18. M. Murata, S. St.Laurent, and D. Kohn. XML media types, January 2001. <ftp://ftp.rfc-editor.org/in-notes/rfc3023.txt>.
19. P. Topping. Mathml requirements. Presented to the "XML in HTML" Coordination Meeting, Feb 1998, see <http://www.w3.org/Math/W3CDocs/mathmlrequ.html>.
20. E. Weisstein. Mathworld, the web's most extensive mathematics resource. See <http://mathworld.wolfram.com/>.
21. B. White. Bringing mathematical formatting to the web, December 2002. Presentation at the InterLab 2002, DOE Laboratory Web Development and Management Conference (<http://www.nrel.gov/interlab/interlab02/program.html>).