TECHNOLOGIES FOR TEACHING MATHEMATICS VIA THE WORLD WIDE WEB

K.E. JUNQUIERA

ABSTRACT

This paper tries to find answers to the question concerning the availability of suitable technologies to accommodate the teaching and learning of mathematics by means of the World Wide Web. It addresses three standards for the presentation of content mark-up and touches on the importance of adequate browser applicability with respect to *MathML* as one of the standards. Various tools for rendering *MathML* on the web, as well as plug-ins and extensions and other combinations of technologies, are discussed. The paper concludes with the introduction of a dynamic mathematics object model (DMOM) by Robert Miner from Design Science Inc. Requirements for a DMOM are formulated and its implementation is discussed.

KEY WORDS

- ***MathML*:** Mathematical Mark-up Language.
  - **Content *MathML*:** Mark-up focused on encoding the semantics of an expression.
  - **Presentation *MathML*:** Mark-up concerned with the way the mathematical expression looks.
- ***OMDOC*:** An Open Mark-up Format for Mathematical Documents.
- **XML:** Extensible Mark-up Language. A Meta language based on SGML that may be used to create other mark-up languages.
- **XHTML:** Extensible Hypertext Mark-up Language. A reformulation of HTML 4 as an XML application.

1. INTRODUCTION

This paper reports on web technologies that are available and can be used to facilitate the teaching of the subject mathematics by means of the World Wide Web. The information was obtained as part of a doctoral study performed at the Central University of Technology, Free State during 2005. As the web presentation of a variety of subjects, including mathematics, is a reality at higher education institutions, the task at hand for mathematics lecturers is to equip themselves with the knowledge of how to use these technologies to the best of their ability. The use of the web in teaching and learning is rapidly becoming the norm and no longer the exception and can therefore not be ignored by any educator at a higher education institution who sees himself as being on the forefront regarding the use of modern teaching and learning methods.

2. CONTENT VERSUS PRESENTATION MARK-UP AND MATHEMATICAL NOTATION

Before we can enter into a discussion of the available web technologies, a few words have to be said about the nature of mathematical notation and language. The differences between content and presentation mark-up has to be noted as well.

When studying the technologies needed to present and learn mathematics via the World Wide Web, a clear understanding of the notation involved is essential. Mathematical notation is not just a set of symbols and signs that mathematicians use. It is a language that has arisen through a historical process just like any spoken language. “But now the question is: can computers be set up to understand that notation?” (Wolfram, 2002: Computers: 1 of 13). This question is of the utmost importance. If computers cannot understand the mathematical questions put to them by students using the mathematical language, they will not be able to assist these students in their acquisition of knowledge in this field.

According to Wolfram (2002), there are two sides to using mathematical languages. They are called the input side and the output side. On the output side the situation is straightforward. This is the side on which the mathematician, the lecturer or the student does the typing of mathematical text. Mark-up languages such as *MathML* and *TeX* or *Latex*-applications such as *Scientific Workplace* and *PCTex* have been developed to facilitate this need. The problem, however, is not so much on the output side as it is on the input side. On the input side the computer needs to interpret what has been typed and then respond to it. Consider the following expression, for instance:

$$\arctan[x-1]\text{^}2+\sin[x-1]+c(x-1)-g[x-1]$$

The expression contains mathematical text with operands and operators. How does the computer now tell what groups with what? It will only be able to identify groupings if it has the knowledge of the precedence of the operators. It therefore implies that the computer should know which operators bind tighter to their operands (Wolfram, 2002). In the above expression, for example, the constant c binds tighter through multiplication to $(x-1)$ than it does through addition to $\sin[x-1]$. The use of templates of free-form input as used by Wolfram Research in *Mathematica* 3.0, attempts to solve this problem on the input side.

It is evident that notation constitutes the first stumbling-block that has to be overcome in an attempt to present mathematical information successfully to students via the web. A second has to do with the communication of mathematics between the student and the programme that carries the learning content.

The computer and the World Wide Web are in essence merely vehicles that accommodate and transport this learning content. In an attempt to solve the computer-related communication problem a mathematical mark-up language, *MathML*, has been developed. This mark-up language addresses the notational preferences and symbolic ambiguities of mathematical communication by providing two encoding schemes for mathematical objects and by defining a mechanism for binding the one encoding scheme to the other (Huerter, Rodionov & Watt, 2002). These encoding schemes are called semantic (or content) encoding and notational (or presentation) encoding. Not all *MathML* applications use the same type of mark-up, however. Different uses of *MathML* encoding necessitate transformations between the content and presentation encoding, in order to share encoded mathematical objects successfully among different types of *MathML* applications. Extensible style-sheet language transformations (*XSLT* style-sheets) are used to transform *MathML* documents from one encoding scheme to another.

3. ADDRESSING CONTENT MARK-UP ON THE WEB

"Currently, almost all mathematical documents available on the Web are marked up only for presentation, severely crippling the potential for added-value services like concept-oriented navigation and information retrieval, data mining or document personalization" (Kohlhase & Asperti, 2002: 1 of 7). The absence of content mark-up in mathematical web documents restricts the possibility of interaction between the student and the mathematical content, as presentation mark-up is non-interactive. To obtain added-value-services and interactivity, preference has to be given to content mark-up for publishing mathematics on the web. It involves a passing from machine-readable to machine-understandable representations of mathematical information.

According to Kohlhase and Asperti (2002), the World Wide Web is the largest single resource of mathematical knowledge, and its importance will grow as display technologies like *MATHML*, *OPENMATH* and *OMDOC* emerge. These three display technologies that assist with the managing of aspects of mathematics teaching via the World Wide Web, are subsequently discussed.

3.1 *MathML*

The *MathML* standard was started with the vision of addressing all possibilities with respect to mathematics on the web. It was the first web standard that introduced a content mark-up layer in parallel with a presentation mark-up layer, and has therefore become a pioneering project in the field (Kohlhase & Asperti, 2002).

MathML focuses primarily on mathematical expressions, though. In order to bring the idea of a semantic web of mathematics to its full potential, which would be contained in the content mark-up, other layers of mathematical information should also be considered. This information includes a clear mark-up description for proofs; a mark-up for mathematical statements such as theorems, lemmas, corollaries and examples; a mark-up for structured collections of objects such as documents and theories; mark-up for possible relationships between these collections and, finally; a good metadata layer (Kohlhase & Asperti, 2002).

Asperti, Padovani, Coen and Schena (2000) support the choice of *MathML* as a standard for writing mathematics. They justify their choice on the grounds that *MathML* has the following characteristics:

- *MathML* has been conceived as an extensible and thus potentially infinite language.
- Many specific logical *MathML* dialects can be mapped into the same intermediate language or into suitable extensions of the language. Similarly, the intermediate representation can be transformed into different presentation formats.
- Being a standard, *MathML* may be exploited to cut and paste expressions from one application into another.
- Content *MathML* can precisely capture the informal semantics or meaning of well-known operators, such as equality, for example.

3.2 *OPENMATH*

OPENMATH is a standard most suitable for the content mark-up contained in mathematical expressions and its focus is on extensibility. Instead of supplying a wide range of primitive elements representing mathematical concepts and operators, *OPENMATH* totally relies on a representation of primitive symbols that reference concepts defined in so-called content dictionaries. These dictionaries are *XML* documents in a standardised form that define symbols and specify their meaning. *OPENMATH* supplies symbols from content dictionaries for all *MathML* primitives, for content *MathML* and for *OPENMATH* itself. An example is the csymbol element. As a *MathML* element it can provide the same function as the *OPENMATH* OMS element. *OPENMATH* and content *MathML* are therefore roughly isomorphic (Kohlhase & Asperti, 2002). From the above is it evident that many *OPENMATH* and content *MathML* characteristics overlap and correspond.

3.3 *OMDOC*

The *OMDOC* format integrates content *MathML* and *OPENMATH* with respect to mathematical expressions and extends the combination on the level of mathematical statements, including definitions, lemmas and proofs, as well as on the level of mathematical theories (Kohlhase & Asperti, 2002). This integration allows better navigation, theory re-use and modularisation, specifically on theory level. Even though standardisation efforts for theories in the field of algebraic specification have been made, *OMDOC* is the first real result from efforts to develop the combination of *MathML* and *OPENMATH* into a general mark-up language for mathematics, with attention to web communication and existing standards.

4. THE USE OF STYLE-SHEETS

The development of *MathML* goes back to the early days of the Web itself. Years of vigorous debate and search for consensus in ways to publish mathematics through the web finally led to the first *MathML* recommendation that was announced in 1998 (Sidje, 2002: 1 of 2). The fact that mathematics could then be written successfully by using technology did not necessarily mean that it could be successfully reproduced and read from a computer screen by those interested.

To read mathematical content via the web, a browser is needed that can clearly and unambiguously reproduce the coded *MathML* as mathematical text that conveys the original meaning intended by the author. Teaching mathematics via the web would therefore be impossible without an applicable browser. But a very limited few of the current generation of web browsers support *MathML* by default. Initially it even appeared as if traditional browser makers were not in any particular hurry to invest their resources into integrating support for mathematics (Sidje, 2002). As a result *MathML* content did not spread very much on the World Wide Web initially.

Until recently authors of documents that contain *MathML* needed to place specific mark-up in the document to enable the *MathML* rendering. This specific mark-up committed the document to being read by only one particular browser. It was a situation far removed from the ideal of posting documents with mathematics content on a web server to be read by anyone irrespective of the browser that they used (Carlisle, 2002). Over recent years there have been efforts that today are resulting in readily available means to represent *MathML* in the major browsers. Nowadays specified extensions have been developed that enable good quality *MathML* rendering in a surprisingly large set of browsers or operating system combinations. One such an extension type, which aims at greatly reducing the need for rendered-specific mark-up to appear in the document, is an extensible style-sheet, *XSLT* in short. The style-sheet, named the Universal Math Style-sheet, and the document should operate from the same server, however. The author only needs to link the document to the Universal Math Style-sheet. The style-sheet will detect the browser and the type of *MathML* mark-up that is used and will transform the document within the browser, adding whatever extra mark-up is required to make it readable (Carlisle, 2002).

XSLT (extensible style-sheet) is a language designed to transform *XML* (extensible mark-up language) documents (Carlisle, 2002). Since *MathML* falls into the extensible mark-up languages (*XML*) category, *XSLT* is applicable to documents that were written by using *MathML*. We can now conclude that, "…*MathML* is now ready to be used on the web… There are freely available robust implementations of *MathML* renderers [*sic*] which work with the current generation of browsers" (Carlisle, 2002: 3 of 3). This is good news, as it ensures that mathematical text written in *MathML* will be able to be rendered on the web with the use of an applicable browser. In the next section we consider such browser applicability.

5. BROWSER APPLICABILITY WITH RESPECT TO *MathML*

Different browsers have to be considered in an attempt to determine how successfully they operate with respect to *MathML*. According to Quint and Vatton (2002), a technology such as *MathML*, although necessary since it can be used to publish mathematics on the web, is only the so-called tip of the ice berg when teaching and learning by means of the web is considered. *MathML* should be implemented in tools that students can use easily. Such tools should be able to display web pages with equations and provide complementary services that facilitate the various tasks involved in the teaching process (Quint & Vatton, 2002). Students should therefore not only be able to see and read mathematical material from the web, but should also be able to create their own online documents or to complete documents written by lecturers. A complete web client must provide powerful editing functionalities through an easy to use browser interface. In such a situation students will be able to focus on the actual content of the document that they are compiling, instead of struggling with the tool (Quint & Vatton, 2002: 2 of 5).

Just as writing with a pen comes naturally, using the editing tool should become a basic skill. This requirement implies that entering mathematical expressions in textual notation is not an option. Students should not have to learn such a textual language in addition to learning the mathematical language and concepts covered in a syllabus. A direct manipulation style of interface is therefore required. The interface should be as simple as possible, providing only the commands the user needs and can understand. To this extent, even a what-you-see-is-what-you-get interface may present an important cognitive load for a beginner student. Furthermore, the tool should be flexible enough to allow lecturers to use the same tool in various classes (Quint & Vatton, 2002).

Another important aspect contributing to ease of use is consistency. Writers of mathematical documents, such as lecturers and students, do not manipulate equations in isolation, but rather do so within documents that also contain text and graphics. It is therefore important that a uniform environment exists in which all the content can be manipulated in a consistent way (Quint & Vatton, 2002). The above-mentioned requirements underline the need for a tool that would be customisable.

As a browser, *Mozilla* natively provides *MathML* support and contains the source code for *Netscape 6.x* (Sidje, 2002). *Amaya* is yet another tool that was designed while taking most of the aforementioned requirements into account. It is a full-featured web client that seamlessly integrates editing and browsing functionalities (Quint & Vatton, 2002). Some of *Amaya's* most important functionalities are the following: *Amaya* can display and print web pages that are stored on local files or that lie on web servers.

It can edit those pages or create new ones. *Amaya* can save documents either locally or on remote web servers, whilst from the user's point of view, there is no difference between documents stored in the local filing system or elsewhere.

Amaya closely follows the *MathML* 2.0 specification for editing equations within web pages. As the primary goal is to allow users to easily enter complex expressions, the emphasis was put on presentation *MathML* (Quint & Vatton, 2002). Although content *MathML* was not implemented, all the presentation tags and attributes are available. The usage of these tags and attributes leads to a very natural user interface that could be compared to the way that mathematics is written by hand.

Using presentation mark-up means that the user is not required to enter all the details of the *MathML* structure (Quint & Vatton, 2002). The editor generates elements on a lower level of interpretation automatically. It is based on simple heuristics that recognise numbers, operators, function names, etc. When the representation is not linear, however, the user needs to enter those commands that generate the main constructs of presentation *MathML*. Palettes containing these constructs can easily be created and used when only mathematics is edited.

In summary, it is clear that however big the contribution of *MathML* to the publishing of mathematics on the web might be, it becomes useless to both students and lecturers if it is not incorporated into an application that supports both browsing and editing functionalities. *Mozilla* and *Amaya* were mentioned as suitable browsers that accommodate *MathML* and which could be utilised in the teaching of mathematics by means of the World Wide Web.

6. PLUG-INS AND EXTENSIONS

Most tools for rendering *MathML* are either full web browsing applications, such as *Amaya* and *Mozilla*, or plug-ins and extensions, such as the *WebEQ* applet, *IBM Techexplorer* and *MathPlayer*. In the first case, the *MathML* formatting engine is implemented as part of a more general layout engine used for *HTML* display, which permits *MathML* fragments to integrate within *HTML* mark-up. In the second case, an external component is embedded into a larger browsing application by means of an application-specific extension mechanism. In such a case the *MathML* formatting engine is completely separate from the general layout engine and *MathML* fragments are rendered inside rectangular portions of this rendering area (Padovani, 2002). A number of well-known plug-ins and extensions that could be used to render *MathML* will now be discussed.

6.1 THE *IBM MathML Expression Editor*

The *IBM MathML Expression Editor* is an example of an external *MathML* application that can be used for rendering, authoring, converting, and interacting with expressions that are encoded by means of content or presentation *MathML* elements (Dooley, 2002). It consists of a standalone application edition as well as a component edition. Both editions use the same core framework for editing *MathML* elements and builds on the rendering support for *MathML* presentation elements provided by the *IBM Techexplorer Hypermedia Browser*. This browser provides mathematical editing facilities for the entire range of *MathML* 2.0 content and presentation elements (Dooley, 2002).

The standalone application edition of the editor provides a compact tool for creating *MathML* expressions that can be incorporated into web pages or other documents. These expressions can then be rendered in a variety of containers using the component edition of the editor, since it supports the same editing functionality as the standalone application edition. *MathML* elements that are embedded in other documents become immediately interactive and a user can edit them by using the component edition in the host container in the same way as when the standalone application is used directly (Dooley, 2002).

Because the *IBM MathML Expression Editor* supports both content and presentation *MathML*, these interfaces can be used to convert content to presentation *MathML*, or to incorporate presentation mark-up within content mark-up and vice versa. The focus of the editor is twofold and includes the creation of content *MathML* elements and the production of a co-ordinated *MathML* presentation of those elements.

The *MathML* presentation elements are used only to display the expressions and therefore no direct support for editing presentation *MathML* has been implemented in the editor (Dooley, 2002). Template-based transformations are commonly used for editing content *MathML* elements in the *IBM MathML Expression Editor*. This is done by inserting and removing the desired structures such as elements, attributes and character data.

According to the developers, the programming interfaces provided by the editor bring content-based mathematics-aware applications to the web in the form of a re-usable component that can provide user interaction with mathematical expressions and that can accept user feedback containing mathematical information. In conclusion, the developers of the *IBM MathML Expression Editor* consider it to be a powerful and flexible new tool for rendering and authoring *MathML* expressions that support a high degree of interactivity on deployment (Dooley, 2002). These considerations are fundamentally important as interactivity is the main consideration when teaching mathematics via the World Wide Web.

6.2 THE *WebEQ Developers' Suite*

Another example of an external *MathML* application is the *WebEQ Developers' Suite*. It consists of two main tools for authoring *MathML*, namely an equation editor, the *WebEQ Editor*, and a document translator, the *WebEQ Publisher* (Miner, 2002b). Both tools can be configured to generate *MathML* output, as they were tailored to be used with Design Science's *MathPlayer* application or David Carlisle's *Universal Math Style-sheet*.

The *WebEQ Editor* is a graphical *MathML* equation editor in which expressions are built up by using templates, symbol palettes and keyboard shortcuts (Miner, 2002b). As an equation is edited, a presentation *MathML* data structure is created for it. The *WebEQ Editor* can, however, also be used to generate content *MathML*. These options are possible, as the editor provides a *MathML* export configuration dialogue that allows authors to specify what kind of *MathML* mark-up should be generated.

The *WebEQ Publisher* is a tool that can be used for the hand-authoring of *MathML* documents. It generates an output document from a source document by translating mathematics mark-up in the source into a web-ready format (Miner, 2002b). The *WebEQ Publisher* recognises both *MathML* and *WebTex* mark-up in the source and can generate *MathML*, images or *WebEQ ViewerControl* applet wrappers. The publisher can also be configured to add header declarations for *MathPlayer*, David Carlisle's *Universal Math Style-sheet* or for *Mozilla* if they are needed.

6.3 *MathPlayer*

"*MathPlayer* is a state-of-the-art plug-in that enables *Internet Explorer (IE)* to display *MathML*" (Miner & Topping, 2001: 7). Developed in collaboration with *Microsoft* using *Internet Explorer's* behaviour or binding technique, *MathPlayer* seamlessly integrates *MathML* into web pages. Behaviour technique is a new plug-in technology from *Microsoft* that allows much better integration into web pages than other plug-in technologies. *MathPlayer* can automatically match font sizes and styles, as well as align mathematical input with surrounding text. The use of *MathPlayer* results in high quality printing. It is designed to be part of the *HTML* Platform and can be scripted with *JavaScript* and styled with *CSS*.

6.4 *MathType*

MathType 5 is the next version of Design Science's first *MathType* equation editor (Miner & Topping, 2001: 7 – 8). It involves *MathPage* technology and revolutionises the process of exporting a *Microsoft Word* document with mathematics in it to the web. Design Science Inc.'s *MathPage* technology augments *Microsoft Word's* "Save as *HTML*"-feature in powerful ways, as *MathPage* uses the full power of the *HTML* platform to create *HTML* pages with mathematics as *MathML* instances that can be displayed by *MathML*-capable browsers.

Such browsers include *Mozilla*, *Amaya* and *Internet Explorer* with *MathPlayer*. *MathPage* can also produce mathematics as *GIF* images with high resolution for printing. These mathematics images are all perfectly aligned with the surrounding text (Miner & Topping, 2001: 8). The mathematical islands presented as pictures are non-interactive and therefore do not contribute as positively to the online learning experience as interactive material would be able to do.

6.5 *Maple*

Maple is another system that supports the rendering and editing of *MathML* (Bernardin, McCarron & Harder, 2002). It is a server-based system that is mainly used for symbolic and numeric computation. *Maple* allows the import of *MathML* encoded mathematical expressions. As mentioned in section 5.4 of this chapter, content *MathML* can easily be imported into a computational system like *Maple*. Importing a *MathML* expression with multiple different encodings such as presentation and content *MathML* is also straightforward, as one can choose the encoding that carries the most semantic information and is easiest to correctly import.

Because of the ambiguities of the presented objects, importing pure presentation *MathML* is much more difficult (Bernardin *et al*., 2002). *Maple* resolves this problem concerning presentation *MathML* with a rule-based transformation engine that encodes a fixed set of context dependent choices as to how to interpret elements from presentation *MathML*. “The output of the transformation engine is Content *MathML* with unambiguous semantics, allowing us to reuse our code for importing Content *MathML* in order to do the final translation step to a *Maple* expression” (Bernardin *et al*., 2002: 2 of 5). The imported expressions can then be used in any computation within the *Maple* system.

The resulting expressions of any such computation can be exported back to *MathML* or published to the web (Bernardin *et al*., 2002). Even though *Maple* can export the expressions that it generates to both presentation and content *MathML*, the preferred export format is a combined (parallel) tree containing both presentation and content *MathML*. This export format allows applications that re-use *Maple’s MathML* output to be able to choose the most appropriate format.

Clearly there are currently many technologies that can be used successfully to place mathematical information on the web. Not all of them consist only of *MathML* or *MathML* applications, however, and not all of them support the same degree of interactivity, which is necessary for the successful teaching of mathematics via the web. These combinations of technologies, together with their advantages and disadvantages as set out by Miner and Topping (2001) in their Status Report of Mathematics on the Web, are considered next.

7. COMBINATIONS OF TECHNOLOGIES

Combinations of technologies include *HTML* pages with *GIF* images for equations, *Adobe Acrobat* to read *PDF* documents, *IBM Techexplorer* pages, *HTML* and components such as Design Science's *WebEQ* or *IBM Techexplorer* pages, and server-side programming in applications such as *CGI*, *Perl*, *Java* and *ASP* scripts.

7.1 *HTML* PAGES WITH *GIF* IMAGES FOR EQUATIONS

Because these web pages are *HTML*-based, mathematics can more easily be combined with other media such as movies, sounds, interactivity and data access. They do not require server-side support and there is no need for the installation of fonts, an important consideration for cross-platform browsing.

The most important disadvantage is the fact that interaction with the mathematical content is limited, as the equations are posted as images. There is thus no difference between the mathematics obtained from the web and the mathematical content printed in a textbook. Second, printing such content is not of a high quality, as *GIF* images are more often low-resolution screen images (Miner & Topping, 2001).

7.2 *Adobe Acrobat* (PDF)

PDF documents are effective for the online delivery of information that uses paper as the primary publishing medium. It delivers good quality formatting and printing that are faithful to the designer's original intent. It does not require server-side support and if mathematical fonts are embedded into the document, the mathematics symbol display is guaranteed to work.

Because the format is not *HTML*-based, PDF documents cannot easily be combined with other media such as movies, sound, interactivity and data access. The web browser merely hosts the PDF viewer application, which must integrate its own user interface with that of the browser. This integration can be cumbersome or confusing to the user and limits interaction. Changes made using the browser's interface will have no effect on the PDF display (Miner & Topping, 2001).

7.3 *IBM Techexplorer* PAGES

Herewith mentioned are some advantages and disadvantages of constructing mathematical text by using the *TeX* language together with the *Techexplorer* plug-in. *IBM Techexplorer* pages are written by using the *TeX* language. This language is understood by many mathematicians and is easy to use, as it enables the user to construct mathematical text in a way that is similar to how it would have been written by hand. These pages do not require server-side support and can be printed with *TeX* quality.

To compile *IBM Techexplorer* pages, the *Techexplorer* plug-in is required. Unfortunately the plug-in is not free, except for its trial version that does not allow documents to be printed. Although *Techexplorer* does allow the inclusion of basic media formats, it cannot be combined with the full range of media available in *HTML* pages. Similar to the PDF Viewer, the web browser is merely a host to the viewer application, which must integrate its own user interface with that of the browser. This integration can once again confuse the user and limit interaction. Changes made by using the browser's interface will have no effect on the display (Miner & Topping, 2001).

7.4 *HTML* AND COMPONENTS SUCH AS DESIGN SCIENCE'S *WebEQ* OR *IBM Techexplorer* PAGES

This section mentions some advantages and disadvantages of using ordinary *HTML* for text accompanied by other components for representing instances of mathematics. The fact that the document is *HTML*-based simplifies the integration with other web technologies. The implementation of Design Science's *WebEQ* or *IBM Techexplorer* components supports quick interaction, as it takes place on the client's side and does not require server-side support. Interaction is powerful, as generated scripts can be displayed on the fly.

The interface of the *HTML*-page requires that the width and the height of each individual equation be known in advance. If the mathematics is generated through the component by means of scripting, this information is extremely difficult to obtain beforehand. Mathematical pages containing hundreds of individual equations can cause performance problems due to the inability of browsers to handle that many instances of components. Using components in conjunction with *GIF* images for non-interactive equations can minimise this problem (Miner & Topping, 2001).

7.5 Server-side programming: *CGI, PERL, JAVA and ASP SCRIPTS*

Server-side-programming produces pages that are similar to the ones described in the above section on *HTML* and components. Such programming can avoid image size problems and results in pages with the widest range of browser compatibility. The load on the client, that is the browser, is minimal because all scripts are executed on the server and scripting only needs to target the server platform and not all the browser platforms. Server-side-programming works well for accessing specialised software such as computer algebra systems.

Server-side scripting is difficult to write and debug as the programming is done on a different computer as the one from which the browser operates. Furthermore, the scripting must serve multiple clients simultaneously and runs the risk of making a production server unstable. Unlike other solutions, the whole programming load is concentrated on the server.

User interaction furthermore requires a visit to the server, which makes interaction sluggish and unpredictable (Miner & Topping, 2001).

The choice as to which of the above technologies should be implemented is once again determined by the availability of the specific technology to its user and by institutional policy. As each of the described technologies have advantages as well as disadvantages, no recommendations as to which is most applicable to the presentation of mathematics via the web, can be made. According to Miner and Topping (2001:4), however, “Each of the many technologies for putting math on the Web has its merits. However, in the long term there can be little doubt that, taken together, the collection of HTML-centric technologies that have taken shape through years of industry collaboration at the W3C [World Wide Web Consortium] represent [sic] the future of the Web”. This collection of web standards is called the *HTML* Platform.

8. THE DYNAMIC MATHEMATICS OBJECT MODEL

“Creating dynamic mathematical content for the Web is difficult” (Miner, 2002a: 1 of 6). The reason for this observation is simple: the authoring of static content does not involve difficult programming aspects that are inherent in the authoring of dynamic content. The situation is further exacerbated by the lack of an effective model for the programming of dynamic mathematical content (Miner, 2002a). Although a great number of technologies are available for placing mathematical text on the web, there is no single recommended model that can effectively be used by mathematics lecturers in all situations to place dynamic mathematical content on the web. All the information that has been provided so far with respect to the teaching of mathematics by means of the web and the implementation of corresponding technologies is now brought together by the introduction of the object model concept.

A dynamic mathematics object model (DMOM) is a mathematics application-programming interface (API) that can be used to provide programmatic access to an extensible mark-up language and in doing so enable mathematics lecturers to place mathematical content on the web. DMOMs can roughly be divided into two categories. The first category contains DMOMs that primarily use the web as a delivery vehicle for information (Miner, 2002a). In this category the interactivity is confined to the rectangle of the plug-in, applet, or the control. *LiveMath* and *MathWright* are two commercial products that provide systems for authoring and displaying dynamic mathematics in web pages. *Cinderella* and *Geometer’s Sketchpad* are specialised dynamic mathematics authoring tools that can also publish to the web. *Simulink* from the *MathWorks* offers another kind of DMOM, emphasising visual programming. These simulations cannot be published to the web, however.

The second category of DMOMs contains mathematical web services (Miner, 2002a). *Maplets*, from Waterloo Maple, provide a client-side user interface for back-end services from a *MapleNet* server. Similarly *WebMathematica* from Wolfram Research provides the computation back-end for a server page model. J/*Link* technology provides a *Java* bridge for client-side user interface applications, whereas the *Matlab* web server from *MathWorks* offers a more traditional *CGI*-style solution.

Unfortunately, most of these models in the second category of DMOMs emphasise server-side computation (Miner, 2002a), while server-side computation is not what is needed for interactive, online mathematics education where lecturers are the ones posting the content onto the web. In addition, web interactivity is added as an extension to existing desktop software in all of the cases mentioned. But extensions and plug-ins become sluggish to work with and are therefore also not the ideal solution.

“However, there are also a number of dynamic math object models implicit in the many excellent dynamic math web sites created over the last five years” (Miner, 2002a: 2 of 6). Two such models that are worth mentioning are *Project LINKS* and the *ActiveMath* Project. These sites provide an excellent guide to what the requirements for a dynamic mathematics object model (DMOM) should be. In the next section, requirements for such a DMOM are formulated.

8.1 Formulating requirements for a dynamic Mathematics object model

The formulation of requirements for a DMOM tailored for the World Wide Web can now commence. A list of five common dynamic mathematical tasks, that the model should be able to perform, is considered first of all (Miner, 2002a). The DMOM should:

- be able to step a student or reader through a certain number of explanations to a given problem;
- enable a student or reader to change parameters to observe the effect on a given equation;
- enable a student or reader to manipulate equations to observe the effect;
- be able to display typeset previews of encoded input, like graphs and tables; and
- be able to obtain and check answers to online test questions.

According to Robert Miner (2002a) from Design Science, Inc., four broad kinds of functionality on the side of the DMOM are involved in accomplishing such tasks as stated above. These functionalities are:

- The manipulation of mathematical notation

The DMOM should be able to modify an arbitrary equation in mathematically natural ways.

- Operations on mathematical expressions

The DMOM should be able to check two expressions for equality, check an expression for a pattern match, evaluate and compute with expressions, and convert expressions into other formats such as images.

- Working with mathematical-aware user interface widgets

In this respect the DMOM should be able to display an equation in a browser that does not support native rendering capabilities. It should allow a student or reader to modify an existing equation in prescribed ways such as filling in the exponent or taking the root. It should be able to obtain and understand free-form equation input from a student or reader and should be able to display an interactive graph.

- Managing interaction with students

Expressions should be able to respond to mouse gestures and controls that are operated by the student. To initialise the mouse and the controls, they should be manipulated programmatically.

In designing a dynamic mathematics object model the goal according to Miner (2002a) should be to make the performance of these four categories of tasks by the DMOM as easy and natural as possible within the existing web development paradigms.

8.2 A dynamic mathematics object model from design science inc.

At Design Science Inc. a Dynamic Mathematics Object Model proposal was made and a prototype implementation targeting *Internet Explorer 6* was carried out. At the 2002 *<math>ml and Technologies for Mathematics on the Web* International Conference, hosted by Wolfram Research, Robert Miner discussed the proposal and its prototype implementation, which will be referred to in this section. The fairly simple, Web-oriented DMOM is based on the requirement analysis of the preceding section and consists of three collections of objects: *Equations*, *EquationControls*, and *MathServices* (Miner, 2002a: 3 of 6).

The *Equation* object is in essence a wrapper or cover for a *MathML* expression and provides an interface for both the high and the low level manipulation of the mathematical expression that it represents. In particular, it provides a standard DOM interface for the arbitrary manipulation of its expression. It can also expose other higher-level interfaces for manipulating the expression it represents in more mathematically meaningful ways (Miner, 2002a).

MathService objects perform operations on *Equation* objects. Performing meaningful computations with mathematics expressions and equations is typically the most challenging part of developing dynamic mathematical content for the web. While some simple operations are appropriate for additional client-side implementation, in many cases, computations either require substantial libraries of client-side code, or are best performed on the server-side by specialised software. *MathService* objects provide a layer of abstraction so that mathematical content developers have a uniform interface that can be used on both the local and the remote computation service-sides (Miner, 2002a).

The *EquationControl* objects provide mathematics-aware user interface widgets that extend the native set of primarily text-oriented user interface widgets provided by *HTML*. There are, for example, *ViewerControl*, *InputControl* and *GraphControl* objects. Last, to manage the interaction with readers the *EquationControl* objects implement the DOM Event interfaces so that content developers can use standard Dynamic Hyper Text Mark-up Language (DHTML) techniques for firing and handling events (Miner, 2002a). As stated at the beginning of this section, the dynamic object model prototype as described above was implemented by targeting *Internet Explorer 6*.

Possible weaknesses of the DMOM can be noticed. The lack of cross-platform support and the reliance on *Java* are limitations. By using *JavaScript* proxy objects, some of the cross-platform and cross-technology details can be hidden, though (Miner, 2002a). A potential strength of the approach is that it lends itself to customisable template documents. Many authors who are not capable of creating elaborate dynamic mathematical content from scratch will be able to understand and customise template documents if they keep to standard, well-known web development techniques. Unfortunately it seems likely that even if such an approach was fully implemented, the prerequisite skill level for authors would remain relatively high. Robert Miner (2002a) considered two possible stumbling-blocks with respect to the implementation of the constructed DMOM.

First, the DMOM developed by Design Science and described in this section relies heavily on the availability of *MathML*-aware browsers, browser extensions, stylesheets and *JavaScript* libraries. The worldwide deployment of these *MathML*-aware runtime environments to date has been hampered by the lack of quality content (Miner, 2002a). The problem could be addressed, though, by demanding formal standardisation in industry.

Second, the question must be asked whether a viable authoring community would exist, even if all other issues could be resolved. Experience with the prototype suggested that a relatively high skill level would still be required of authors (Miner, 2002a). This obstacle might even be more crippling to the development of dynamic mathematical content on the web than any other technological obstacle discussed.

9. CONCLUSION

This paper has shown that a range of dynamic mathematics object models exists in addition to the many *MathML*-based programmes and technologies. Examples of such dynamic object models include *LiveMath*, *MathWrite* and *Geometer's Sketchpad*, while examples of *MathML*-based programmes include *MathType*, *MathPlayer*, *Maple* and the *WebEQ Developers' Suite*. All of these technologies can be used to accommodate mathematics on the web, to a certain extent. None of them, however, are completely adapted to mainstream web development techniques.

It was established that the implementation of a single, effective, widely shared underlying model (DMOM) for dynamic mathematics programming, was necessary. By studying existing dynamic object models and websites that use dynamic, interactive mathematics, the staff at Design Science could construct a basic set of requirements for a web-oriented DMOM. A relatively simple DMOM sufficed to satisfy a basic set of requirements and the prototype implementation suggested that such an object model could probably be adequately implemented for mainstream use, notwithstanding all the cross-platform issues. According to Robert Miner (2002: Personal interview), however, this DMOM was not yet ready at that stage to facilitate the successful transfer of new mathematical knowledge to undergraduate students via the World Wide Web. Even though a single dynamic object model for the successful transfer of mathematical knowledge does not yet exist, its absence should not prevent lecturers at higher education institutions from using computer technology in the presentation of mathematics via the World Wide Web.

10. REFERENCES

Asperti, A., Padovani, L., Sacerdoti Coen, C. & Schena, I. 2000. Formal Mathematics in *MathML*. Proceedings from the 2000 conference on *<math>ml and Mathematics on the Web*, USA.

Bernardin, L., McCarron, J. & Harder, D. 2002. *MathML* with *Maple*. Paper presented on 30 June 2002 at the conference on *<math>ml and Technologies for Mathematics on the Web*, Chicago, Illinois, USA.

Carlisle, D. 2002*. MathML* on the Web: Using *XSLT* to enable cross platform support for *XHTML* and *MathML* in current Browsers*.* Paper presented on 29 June 2002 at the conference on *<math>ml and Technologies for Mathematics on the Web*, Chicago, Illinois, USA.

Dooley, S.S. 2002. Bringing *MathML* Content and Presentation Markup to the Web with the *IBM MathML Expression Editor*. Paper presented on 29 June 2002 at the conference on *<math>ml and Technologies for Mathematics on the Web*, Chicago, Illinois, USA.

Huerter, S., Rodionov, I. & Watt, S. 2002. Content-Faithful Transformations for *MathML.* Paper presented on 30 June 2002 at the conference on *<math>ml and Technologies for Mathematics on the Web*, Chicago, Illinois, USA.

Kohlhase, M. & Asperti, A. 2002. *MathML* in the MOWGLI Project*.* Paper presented on 29 June 2002 at the conference on *<math>ml and Technologies for Mathematics on the Web*, Chicago, Illinois, USA.

Miner, R. 2002a. A Dynamic Math Object Model. Paper presented on 29 June 2002 at the conference on *<math>ml and Technologies for Mathematics on the Web,* Chicago, Illinois, USA.

Miner, R. 2002b. Two Ways to Author for *MathPlayer* with *WebEQ.* Paper presented on 29 June 2002 at the conference on *<math>ml and Technologies for Mathematics on the Web*, Chicago, Illinois, USA.

Miner, R. & Topping, P. 2001. Math on the Web: A Status Report. <http://www.dessci.com/webmath/status> Retrieved during February 2001.

Padovani, L. 2002. A Stand-Alone Rendering Engine for *MathML.* Paper presented on 30 June 2002 at the conference on *<math>ml and Technologies for Mathematics on the Web*, Chicago, Illinois, USA.

Quint, V. & Vatton, I. 2002. *MathML* in E-Learning with *Amaya.* Paper presented on 30 June 2002 at the conference on *<math>ml and Technologies for Mathematics on the Web*, Chicago, Illinois, USA.

Sidje, R. 2002. *MathML* amidst Open Web Standards: *Mozilla's* Building Blocks for Today and Tomorrow. Paper presented on 29 June 2002 at the conference on *<math>ml and Technologies for Math on the Web*, Chicago, Illinois, USA.

Wolfram, S. 2002. Mathematical Notation: Past and Future. <http://stephenwolfram.com/publications/talks/mathml/mathml3.html> Retrieved on 14 March 2002.