An Approach for Interoperable and Customizable Web-based Mathematics Education

David Chiu
Department of Computer Science and Engineering
The Ohio State University
chiud@cse.ohio-state.edu

Paul S. Wang
Department of Computer Science
Kent State University
pwang@cs.kent.edu

Abstract

Classroom trials at Kimpton Middle School demonstrated advantages of the WME (Web-based Mathematics Education) system and obtained positive feedback as well as suggestions from teachers and students. Based on the prototype site for Kimpton, an interdisciplinary team is developing a model WME site making it directly deployable in different schools. Components within WME sites are interoperable and easily customizable by teachers. Reported are the model site design, organization, architecture and implementation. Features that support portability, component interoperability and easy user customization are emphasized. These features are important in making WME practical, efficient, and effective for mathematics education.

1. Introduction and Background

Students in the US are falling seriously behind other countries in mathematics tests. Furthermore, recommendations for mathematics education by distinguished groups of scientists and educators have been for the most part ignored [8]. According to [10], ``The fastest growing jobs require much higher math, language, and reasoning abilities than current jobs, while slowly growing jobs require less." It is clear that students need to master mathematics more deeply and at higher levels than they do at present.

The increased curricular demands on students mean mathematics teachers are required to know more mathematics themselves. This coupled with the push for high stakes tests has caused shortages in highly qualified mathematics teachers. The amount of cognitive, organizational, and emotional work needed to be a successful mathematics teacher, already high, is growing. What can be done to support mathematics teachers and help them become more efficient and productive? Can we find a practical and effective way to use modern technology to assist the teaching and learning of mathematics?

At the Institute for Computational Mathematics, an interdisciplinary team of mathematicians, computer scientists, education researchers, Web developers, and middle school teachers is building a *Web-based Mathematics Education* (WME) system by an innovative combination of open Internet technologies.

Figure 1 illustrates the WME concept. WME can deliver, via the Internet or a LAN (wired or wireless), classroom-ready lessons that are well-prepared, interesting, effective, as well as interoperable. In addition to allowing multimedia content and hyperlinks, lesson pages feature in-page *manipulatives* to help students understand and explore mathematical concepts and skills through hands-on activities.

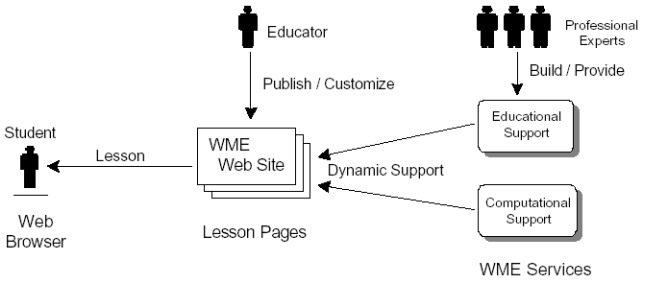

Figure 1: The WME Concept

WME is different from existing approaches and aims to be a modern, practical, efficient and effective Web-based system to support mathematics education and learning [6]. The WME system conforms to open standards, works with regular browsers, delivers integrated and complete lessons, enables easy customization, provides systematic access to client-side and server-side support, and allows independently developed WME components to interoperate seamlessly. In short WME seeks to create a *Web for Mathematics Education*. Figure 2 (top) illustrates some of the contents and services WME can combine and integrate for effective Web-based mathematics education.

1.1 WME Pilot Site

With funding from the Ohio Board of Regents (OBR) the ICM group began, in early 2004, a pilot project to study how the WME technologies can best be tailored and applied for actual mathematics education in schools. The pilot project at Kimpton Middle School (Munroe Falls, Ohio) conducted in-class trials at the 7th grade with two

mathematics teachers and over 100 students that spanned several classes.
Experiences, feedback, and early lessons learned have been reported in [5].

Encouraged by the positive reactions from both students and teachers, we proceeded to build a prototype WME site (Figure 2 top) to implement new ideas and improvements, to teach a variety of additional topics, and to perform in-class trials. The topics include *integers*, *percentages*, *proportions*, *length and area*, *number relations*, *fractions*, *probabilities*, and *understanding data*. These are some of the topics in the *Ohio Academic Content Standards* [11] which closely follows the NCTM standards [9]. These trials have continued and are still on-going.

Figure 2: WME Integration (top), Kimpton Pilot Site (bottom)

Component interoperability and easy customization by teachers are among the most important features of WME. The goal is to provide simple download and installation of WME sites for schools and to give teachers the ability to modify and tailor many parts of the classroom-ready materials to suit their own teaching needs, if necessary. This is illustrated in Figure 3. It is furthermore possible for teachers to publish back to the WME system their improved lesson pages in order to share with others.

We begin by presenting the design and organization of the *WME model site*, site portability, and component independence. Then, our approach to achieve interoperability and customization of WME components will be described.

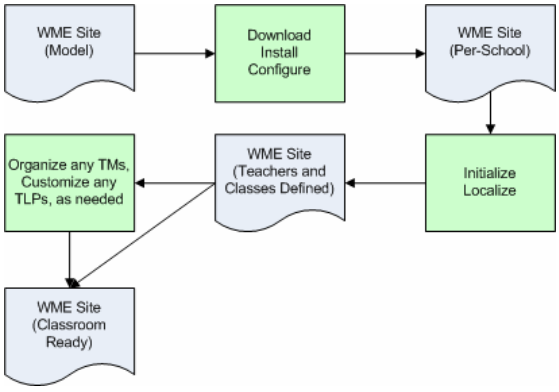

Figure 3: WME Site Deployment

2. WME Components and the Model Site Design

WME interoperability aims to make many of its inner components easily deployable into or removable from any WME system. Our approach to supporting this interoperability relies on hierarchically organizing WME into self-contained components, making them maximally independent of other components while explicitly defining their connections and dependence on related components. Such plug-and-play components are also made customizable by exposing their flexibility for changes. Our reasoning and conclusions leading to the current WME component structure were reported in [1, 3, 4].

A complete copy of the model site is easily downloaded and installed on any standard Web server.
Through data entry (who are the teachers, what courses are taught, etc.), it is further refined and tailored to a particular school. The model site contains two parts:

- Site administration and operations support---These include
  - Modifying school and WME settings
  - User management
  - Course management
  - Topic module management
  - Lesson plan management for teachers
- A detailed description of these is given in the last section regarding customization.
- Educational content---These are organized as topic modules (TM) under one directory. Each TM contains a sequence of lesson pages (TLP) and each TLP focuses on teaching a particular mathematical concept or skill.

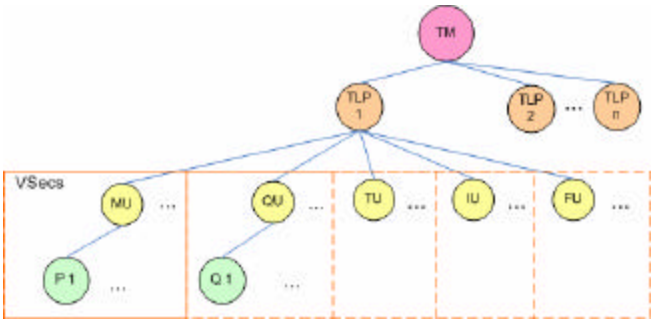

Figure 4: WME Component Breakdown

Following the breakdown in Figure 4, at the root of the WME component hierarchy lays a TM. A TM is similar to a chapter in a book. It covers a topical area (such as integers or percentages) and includes an ordered sequence of interactive lesson pages, or TLPs (think of these as sections in a chapter). TM interoperability means any TM can be installed (or removed) easily and guarantees that the TM will work with the host site seamlessly and becomes immediately available for teachers. There is no way to tell the newly installed TM from the TMs already there at the site. Of course, removal of any TM will not affect the normal functioning of the rest of the WME site.

Similarly, a TLP is a self-contained unit that helps teach a particular mathematical concept or skill, strengthen the student's understanding through hands-on activities, manipulation of tools, games, answering questions, or other interactions within the page. From an instruction standpoint, TLPs are non-linear. That is, a teacher can choose *where and when* to use a TLP without predetermined constraints. Each TM may come with a large set of TLPs relating to a particular area covered by the TM and with a suggested typical sequence of TLPs to use for the classroom. But, a teacher can easily change the sequence, add/remove
TLPs, or introduce a TLP from another module. TLP interoperability states that a TLP can be *dropped into a TM* at any time and its availability for use will become immediately apparent. Removal of TLPs from a TM works as expected.

The lowest tier is the most complex. TLPs are unit-based, but before we venture into details, it is important to note that TLPs are an ordered concatenation of View Sections (VSec). A VSec controls the display visibility of *page units* included in it. Through VSec's, teachers can control what students see in real time, to focus their attention or to avoid distractions. Currently, we have the following five types of page units.

- Text Unit (TU) --- A single Text Unit is a block of plain text or markup (most likely XHTML).
- Image Unit (IM) --- An Image Unit is a discrete graphical or animated file.
- Formula Unit (FU) --- MathML code is contained within each Formula Unit in order to display mathematical formulas naturally and to store them structurally.
- Question Unit (QU) --- A Question Unit provides a *set* of questions to be posed for student response.
- Manipulative Unit (MU) --- A Manipulative Unit is a hands-on exercise often designed as a game to help communicate a mathematical concept to a student visually and interactively. Like Question Units, manipulatives themselves contain inner elements. These are parameter files, that, when supplied to a manipulative, can affect its display, computation, and outcome. We will delve deeper into the architecture and inner workings of manipulatives in the next section.

The purpose of this unit organization is to make it possible for teachers to tailor/edit TLPs in simple but important ways including rewording, modifying mathematical formulas, replacing certain images, adjusting parameters to manipulatives, and adding questions in the right places on a lesson page to collect student answers. Interoperability at this level includes the ability to add, remove, edit, and replace any unit in a TLP.

A direct advantage of WME interoperability is the easy customization of components by teachers to suit their own situations and needs. But, care must be taken because it is unlikely that one teacher's modifications will apply well to all others. Therefore, all customization must be recorded per class and teacher. Later, we will discuss how customization is achieved within WME components.

The hierarchical structure has an underlying advantage in that, as long as component nodes are self-sufficient, they can be added under their parent node for interoperation. Because TMs are modules that can be dropped in and out of any WME system, the model site anticipates a standard file structure for TMs. Due to this sensitivity, it is without question that their data structure must be followed rigorously upon creation. Figure 5 illustrates the TM file structure. Starting from the TM root directory, we see that each TM carries its own configuration file, *tm.conf.xml*, that contains information that is read upon installation of the TM. Information contained in this file includes its module name, author, list of TLPs, etc. The *index.php file*, *img/*, *css/*, and *script/* directories are meant to contain all necessary files of each type, thereby allowing the TM to be self-sufficient and independent from any WME server resources. Same can be said for the TLP directories. TLPs themselves contain their own configuration file, *tlp.conf.xml*, which contains information about the lesson page such as which VSecs make up its content, and what questions are attached to the VSecs. VSecs are themselves self-sufficient and independent from any TM resources, and so on for manipulatives.

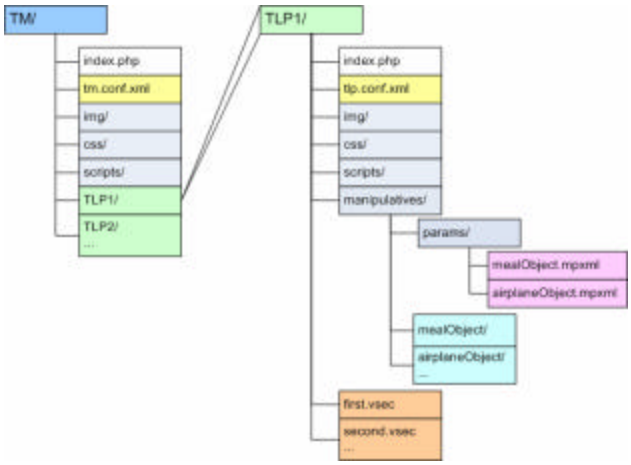

Figure 5: The Topic Module File Structure

3. Manipulative Architecture and Interoperability

Figure 6 shows a manipulative in the *Dinning Out* lesson page inside the *Percentages* topic module. Like many other manipulatives in WME, it is implemented in JavaScript through the Document Object Model (DOM). We will revisit this example time and again in order to illustrate our approach for manipulative interoperability and customization.

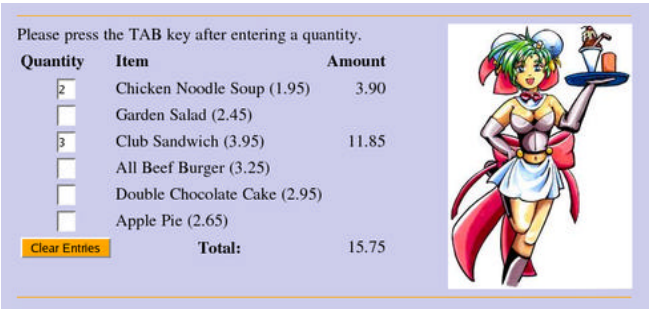

Figure 6: A Sample Manipulative

To achieve interoperability of WME manipulatives we need to make each manipulative a self-contained program and object instance so they are easily deployed in any TLP at any WME site. In fact, a TLP may contain one or more instances of the same manipulative and the same manipulative may be deployed in multiple TLPs. The containing Web page must be able to interact with any manipulative instance through a well-defined interface. Furthermore, each manipulative must be customizable by teachers on a per-instance, per-course, and per-teacher basis. We will describe how JavaScript-defined manipulatives achieve these goals. Other types of manipulatives can follow a similar approach.

Each manipulative is self-contained and has its own directory for all of its files. A JavaScript source file, *MealOrder.js* for example, which defines, among other things, a constructor (MealOrder). The *MealOrder.js* file is loaded into any page that will use the manipulative. To create a manipulative instance, we call the constructor in this way:

```
american = new MealOrder('american');
```

The first argument to the manipulative constructor is always the variable name in JavaScript that holds the instance created. If a second argument is supplied, then it must be an associative array of name-value pairs.
A manipulative instance, such as *american*, is deployed in a Web page by the code

```
<div class="manipulative" id="manip1">
  <script type="text/javascript">
    american.deploy("manip1");
   </script>
</div>
```

In this example, *manip1* is the element id of the * * element where the manipulative is deployed on the page. The class attribute *manipulative* allows us to set up CSS style rules for consistently rendering all manipulatives and easily customizing the style for different WME sites. Each manipulative must define these instance methods:

- **reset()**---Re-initializes the manipulative instance.
- **getArg(***name***)**---Returns the value of the named parameter.
- **getArgs()**---Returns an associative array of the values of all parameters (in the same form as the array used to call the constructor).
- **setArg(***name*, *value***)**---Sets the named parameter to the given value.
- **getProperties()**---Returns an associative array of the values of all instance properties made accessible.
- **getProperty(***name***)**---Returns the named instance property. For example, *total* (the total amount of the bill) is an instance property of a *MealOrder* instance.

Such instance methods can be accessed from anywhere in a Web page for interaction with the manipulative. For example, immediately after the *american* manipulative, the dinning out TLP presents several questions based on the meal order. Figure 7 shows one such question.

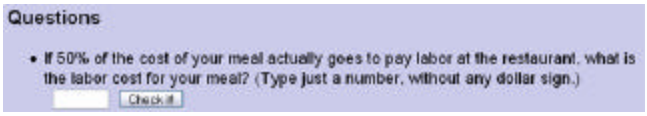

Figure 7: Manipulatived-Based Question

The *50%* part in the question is customizable by the teacher so a different percentage can be used. Its source code is in this form.

```
<span class="pageparameter"
      id="labor_percent">50</span>
```

This editable percentage value and the menu order total are automatically passed to the answer checker via the code

```
american.getProperty('total')

document.getElementById('labor_percent').fir
stChild.nodeValue
```

This architecture allows a manipulative to be easily deployed in another TLP or moved to a different WME site.

4. Manipulative Customization Support

A WME manipulative is made customizable by allowing settable parameters to be passed to its constructor. Therefore, manipulatives can be programmed with much flexibility and generality making them applicable in many more situations. For example, we can make another *MealOrder* instance that displays an Italian menu (Figure 8) by passing to the constructor a menu and a picture as indicated by the following JavaScript code:

```
m = {
    'Italian Wedding Soup': 2.25,
    'Caesar Salad': 3.45,
     'Macaroni and Cheese': 3.45,
     'spaghetti with Meat Balls': 4.65,
     'Tiramisu': 2.95
};

im = "Italian_Waiter.png";
p = {'menu': m, 'image': im};
italian = new MealOrder('italian', p);
```

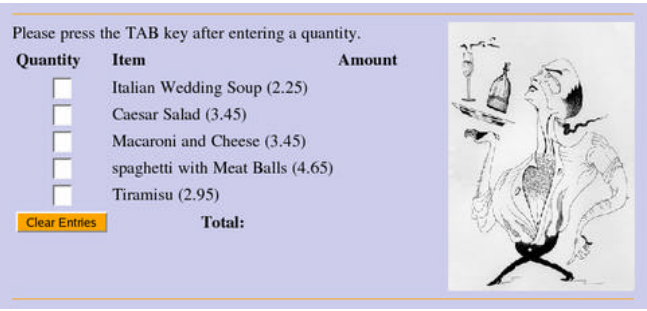

Figure 8: A Customized Manipulative

WME developers (programmers) find such customizations powerful and convenient already. Yet, we still need to allow teachers to customize manipulatives using an easy GUI tool we are developing. The manipulative architecture must support user-level customization as well.

Each manipulative provides these static (class-wide) properties to help usage and customization:

- **instances**---Is an array of ids (strings) for all existing instances.
- **displayType**---Is either "block" or "inline".
- **location**---Gives manipulative directory location relative to site root.
- **document**---Is a text string providing a brief API documentation.
- **help()**---Returns a string for end-user help.
- **help(**"*topic*"**)**---Returns a string for end-user help relating to the given topic.
- **defaultArgs**---Is an associative array for the default arguments when the constructor is called with no args.
- An argument description file in XML---Provides information on parameter name, meaning, allowable values, to help provide end-user customization of constructor arguments. We describe this XML file next, and how it is used to provide manipulative customization on a per instance, class, and teacher basis in the next section.

5. User Customization of WME

On a school-wide scope, every school will have different grade levels, teachers, and even curricula. WME allows these settings to be made on a per-school basis. Another type of customization involves teachers' adjustments to their lesson plans and pages. This section gives an overview of the dual levels of WME customization.

5.1 Level 1 Customization: School-wide Administration

Information that varies across schools is unavoidable. This means that WME must be able to be tailored to every school on this level. In WME sites, there are three types of users: administrator, teacher, and student. Administrators have full control over the WME site. They are allowed the following actions:

- School Settings --- Modify school-specific data such as street address, picture/logo, etc.
- WME Site Settings --- Modify WME site configuration such as database connectivity information.
- Grade Level Management --- Add/Delete/Modify grade levels in this school (e.g. in Kimpton Middle School, we have only 7th and 8th grade levels).
- User Management --- Add/Delete/Modify user accounts. Grant users teacher or even administrator statuses.
- Course Management --- Add/Delete/Modify courses. Constitute what course is to be taught at which grade level and who is to be teaching it.
- Topic Module Management --- Install/Delete TMs for teachers to use in their classes. Administrators, however, have no control over the customization of these modules.

Upon installation of the WME model site, a user account, *root*, is provided. This account is simply for administrators to initiate site settings, and after creation of administrative account, it is recommended that this account be deleted. Like any other system, user accounts with administrative status should be treated with care, because any administrator can compromise the entire site's operation.

5.2 Level 2 Customization: Lesson Management

This level of customization is available only to teachers. In order to expose Level 2 customization in the WME site, the administrators will have already created teacher accounts, grade levels, courses, and installed TMs and TLPs that came with the site. When these are done, the teacher, upon login, will be taken to their administrative area for lesson management. After selecting which course to manage, the teacher can decide which TM to use for that course. At this point the TM will contain original copies of TLPs, and the teacher may decide to use them as-is, or to customize any selected TLPs.

5.2.1 TM Customization

Inside each TM, a teacher can pick and choose from the available TLPs and decide on their sequencing. Thus, the resulting TM will contain only the TLPs picked by the particular teacher and in the desired sequence for teaching the particular class at hand.

Figure 9 (top) shows the TM customization interface for the Statistics module. The TLPs will be listed in the customized TM by the integers their left. To remove a TLP, simply leave the sequence number blank. By following any of the TLP links on the list, the user is transferred to the TLP customization interface.

5.2.2 TLP Customization

After deciding the TLP ordering and usage, the teacher can further customize its content. In the TLP customization interface, a teacher can add content, reword text, reorder and hide VSecs, manage and pose questions, and customize any manipulatives that are instantiated inside the TLP. Figure 9 (bottom) displays a portion of the TLP customization interface. The VSecs (marked by the dark grey dividers) can be reordered, made hidden (or visible) in a TLP.

The modifications by teachers are stored in a per-WME-site database and are dynamically applied to the TLP or TM delivered depending on the teacher and class designations of the user (teacher or student). We use PHP and MySQL for our current sites. But the WME site design does not depend on these particular technologies.

Going even further into page content customization is controlling in-page manipulatives. In the following, we first discuss how manipulative customization is supported, then its interface for editing.

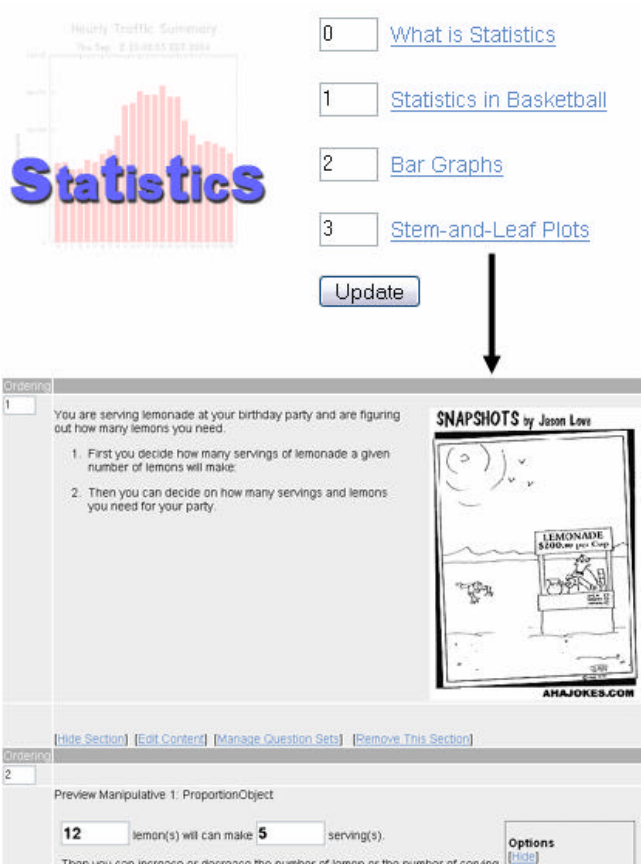

Figure 9: TM Customization Interface (top), TLP Customization Interface (bottom)

5.2.3 Manipulative Customization with MPXML

A manipulative's display and inner computation can be altered on a per-object instance basis by passing various parameters to its constructor. The modified parameter values must be recorded in the site database.

At first, a general colon delimited format was considered for storing these values, but it proves to be not generalized enough. We then begin to consider an XML-defined structure for the job. Other XML applications that can be used to describe source code and parameter data include srcML [2] and WSDL [7], but these seems to be an overkill for our simple purpose of editing and storing manipulative parameter values.

Thus, we developed the MPXML system (Manipulative Parameters XML). This package includes:

- *The MPXML Specification* --- Used to describe manipulative parameters in a way that is generalized and independent from programming languages. Upon manipulative deployment, these values are read and used to instantiate the object instance.

- *Two-Way JavaScript Translator* --- Because MPXML is programming language independent, a parser must be written to tailor to the language of choice (in our case, JavaScript). This two-way parser takes JavaScript code and constructs an MPXML, and in the other direction, takes MPXML data and builds the JavaScript constructor call.

The MPXML definition is straightfoward:

```
<manip_params>
  <!-- The parameter is always an
       associative array -->
  <param type="array">
     ...
     <!-- There can be mixed types within
     the array -->
     <param type="string"> ...  </param>
     ...
  </param>
</manip_params>
```

It is important note here that the <param> element observes more complexity. Its full specification can be found at http://wme.cs.kent.edu/help/docs. Below, the MPXML description to render the MealObject manipulative that we have been following into an Italian menu:

```
<manip_params>
  <param type="array">
    <name>
     <canonical>p</canonical>
     <contextual>
           Manipulative
           Parameters
     </contextual>
    </name>
    <items><!-- first item is the menu -->
     <pair>
      <key> <value>menu</value> </key>
      <param type="array">
       <!-- The parameter, m, is an
              associative array of
              menu items as keys with their
              respective prices as
              values -->
       <name>
        <contextual>
          Menu Items
        </contextual>
       </name>
       <items>
        <!-- First item on menu -->
        <pair>
         <key>
          <value>
           Italian Wedding Soup
          </value>
           </key>
         <param type="double">
          <name>
           <contextual>Price</contextual>
          </name>
          <value>2.25</value>
           </param>
         </pair>
         <!-- Second item on menu -->
           <pair> . . . </pair>
           . . .
         <!-- More items not listed here -->
        </items>
       </param>
      </pair>

     <!-- second parameter, the image -->
      <pair>
       <key> <value>image</value> </key>
       <param type="string">
        <name>
         <contextual>
          Image to be used
         </contextual>
        </name>
        <value> Italian_Waiter.png </value>
       </param>
      </pair>

      <!-- end of associative array -->
     </items>

    <!-- end of parameter definition -->
    </param>
</manip_params>
```

The above will be translated into the equivalent JavaScript just after Figure 8. Note also that nowhere in the MPXML specification was the first parameter, *italian* declared. This is because the first parameter is always implied as the instance variable, which we already know upon instantiation.

The same procedures follow for American meal or any other meal menu that is created by a teacher or another source. As long as the parameter values in the MPXML file are set properly (and this is ensured by the MPXML GUI, which we discuss next), the manipulative will be able to generate customized styles and content for each instance. Figure 10 illustrates the process of deploying a manipulative to a TLP.

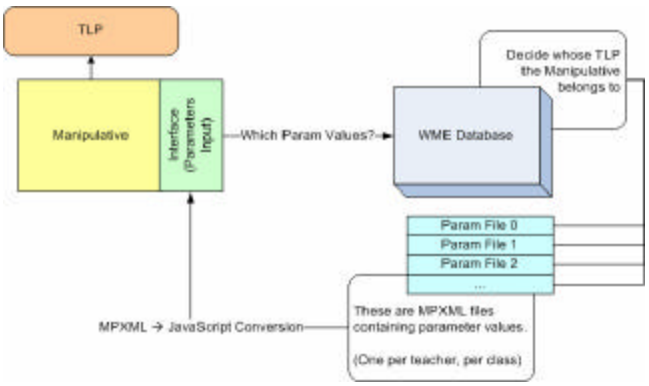

Figure 10: Manipulative Deployment

In order to abate complex workload and guarantee that a valid MPXML and JavaScript constructor call is generated, we cannot leave this task to teachers. We created an intuitive user interface reads the corresponding MPXML file, displays the parameters that can be rewritten, and allows the user to input new values. Before saving, the user has the option of previewing and discarding the values. After commitment to the values, a new MPXML file is written, and the manipulative is reconfigured for this teacher's lesson page only.

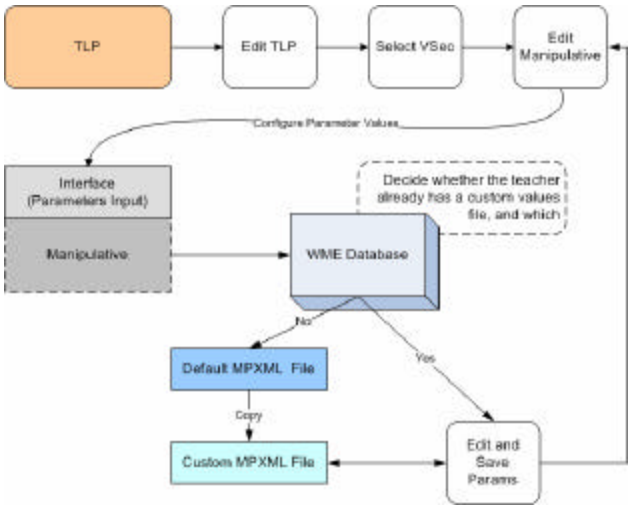

Figure 11: Decisions and Process Involved in Customizing a Manipulative

Figure 11 illustrates the process that the WME site takes during a manipulative modification, and Figure 12 shows the MPXML user interface.

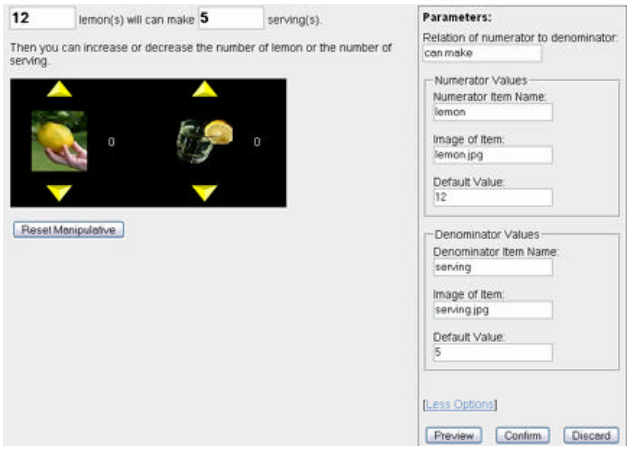

Figure 12: Manipulative Customization Interface

6. Summary and Future Work

Work reported here is part of the team effort by the WME group at ICM/Kent to create a practical, efficient and effective Web-based system to help teachers and students better teach and learn mathematics. Among the distinguishing features of WME are interactive and classroom-ready lessons, easy customization by schools and teachers, and interoperable (plug-and-play) educational components. The goal is to create a *Web for Mathematics Education* that can grow exponentially by enabling many experts and educators to freely contribute to the WME system.

The 1) model site design, 2) the structure, organization and interoperability of components, and 3) the support for easy authoring and publishing of TMs and TLPs by teachers will continue to evolve and refine. Two main tracks of work can be identified: *WME systems R&D* and *educational content and methodology research and evaluation*. The two go hand-in-hand as we continue to drive WME forward.

Through extensive cooperation among an interdisciplinary team and by more extensive classroom trials, we hope to further develop WME to increase its usefulness, usability, and practicality.

7. Acknowledgments

This material is based upon work supported in part by the National Science Foundation under Grant No. 0201772. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Wei Su, Lian Li, and Paul S. Wang. "Lesson Page Structure and Customization in WME," Proceedings of the 2005 IAMC Workshop, Beijing, China, July 24 2005.

[2] Collard, M.L., Maletic, J.I., Marcus, A. "Supporting Document and Data Views of Source Code" Proceedings of the 2nd ACM Symposium on Document Engineering (DocEng'02), McLean, VA , November 8-9, pp. 34-41.

[3] David Chiu. "WME Site Organization and Customization Support," Proceedings of the 2004 Conference on Information Technology in Education (ITE'04), Elizabethtown College, Elizabethtown, PA, September 18, 2004.

[4] David Chiu. "Customization and Interoperability in WME," Proceedings of the IEEE Southeast Conference, IEEE, Ft. Lauderdale, FL, April 8-10 2005, pp. 636-640.

[5] Michael Mikusa, Paul S. Wang, David Chiu, Xun Lai, Xiao Zou. "Web-based Mathematics Education Pilot Project," Conference on Information Technology in Education, Elizabethtown College Elizabethtown, PA, September 18, 2004.

[6] Paul S. Wang, M. Mikusa, S. Al-shomrani, D. Chiu, X. Lai, and X. Zou. "Features and Advantages of WME: A Web-based Mathematics Education System," Proceedings, IEEE SoutheastCon, Fort Lauderdale, Florida, April 2005, pp. 621-629.

[7] Hugo Haas, Team Contact for the Web Services Description Working Group. *WSDL. www.w3.org/TR/wsdl*.

[8] The National Commission on Mathematics and Science Teaching for the 21st Century. "Before It's Too Late: A Report to the Nation," *www.ed.gov/inits/Math/glenn/index.html*, 2000.

[9] National Council of Teachers of Mathematics. *Principles and Standards for School Mathematics*, *www.nctm.org/standards*, Reston, Va., 2000.

[10] National Research Council. "Everybody Counts: A Report to the Nation on the Future of Mathematics Education", National Academies Press, Washington, D.C., ISBN: 0309039770, *http://books.nap.edu/books/0309039770/html/index.html*, May 1989.

[11] Center for Curriculum and Assessment. *Academic Content Standards: K-12 Mathematics*, Ohio Department of Education, Feb. 2002.