Web-based Simulation Architecture for Engineering Education using Java/XML

Wai L. Chan¹, Zhihua Qu², and Issa Batarseh³

Abstract
The objective of this paper is to present a new architecture for designing and developing a web-based teaching enhancement tool for the engineering education. This architecture will enhance student learning by providing an innovative way for them to interact with the common engineering software through the web. Once being finalized, the new architecture will provide a flexible learning environment and capabilities for the engineering students by allowing them to present, test and evaluate their own ideas. To demonstrate the capabilities of this architecture, we will present a simple Java application demo using this architecture. Moreover, we shall discuss briefly a new XML-based markup language, Control Block Diagram Markup Language (CBDML), written during development of this application. Since design of the architecture described in this paper has many possible applications, the CBDML represents the initial result and more work is expected to finalize the architecture and to design and implement various production applications.

1. Introduction

Beside the traditional face-to-face classes, new technologies have been utilized in a lot of university courses to enhance the course material and students' learning experience. Such practice has proven to be very effective for many engineering students. According to the several independence studies, many engineering freshmen and sophomores have difficulty to grasp the basic concepts being taught in the introductory engineering courses. The reason often is they cannot visualize the concepts. Moreover, engineering student may not equip with necessary skill to carry out creative designs using the critical thinking process [5]. Those students, who cannot understand the fundamental principles of the engineering, often fail in engineering courses.

The existing tools, used in these courses, often are stand-alone application and are installed into the computers in the laboratory/computer center. This means the user need to be in front of the computer in order to using them. If the students live away from campus, those students may not have the chance to practice with these tools. In addition to this, the increasing undergraduate enrollment makes these computer resources becoming scarcer. Furthermore, these tools are developed mainly for computation instead of educational use. They can be used for educational purpose but basic training usually is necessary in order that students can utilize the tools.

Over the past few years, several nationwide initiatives have been begun to develop multimedia and web-based educational tools [4,6,7]. Major efforts have been made toward designing and developing Graphical User Interface (GUI) for the static display/presentation purpose. Some efforts focused on developing the web-based simulation tools but only target to a

---

¹ Course Development & Web Services, University of Central Florida, Orlando, FL 32816, USA
² College of Engineering, University of Central Florida, Orlando, FL 32816, USA
³ College of Engineering, University of Central Florida, Orlando, FL 32816, USA
specific design or operation [7]. None has attempted to design and develop a complete set of the software tool for online simulation.

The architecture discussed in this paper is a part of the engineering education enhancement project will change the way we teach the engineering courses at the University of Central Florida. Through this project, a team of faculty members, computer engineering graduate students and engineers will develop fully web-based software packages. It will allow engineering students to be able to design build and simulate engineering problems (such as electrical circuit problems, control system problems, DSP problems, etc.) through a GUI inside of the web browser. This will provide an innovative method for interactive teaching and online design testing. The online computer simulation of the engineering problems will help students better understand the abstract engineering principles and concepts, and will have multiple opportunities to solve engineering design problems [1]. In addition, since the packages will be available through the Web, it is not necessary to install anything in the client side. The students easily have access to the powerful software at home or in the computer laboratory. The software maintenance and upgrade are hustle-free [1].

2. Objectives

The main goal of this paper is to investigate and design a simple architecture that provides a flexible learning environment and capabilities for engineering students either to work through a set of problems posted by the instructor or to present, test and evaluate their own idea through the Web. Hence, it enhances their ability to understand basic concepts and principles through examples, scenarios and exercises.

This architecture would provide the following functionality:

Client
1) Provides GUI to allow the user to establish a block diagram (through which specific models such as differential equations or transfer functions can be input) through the web browser.
2) The client should be able to send a request and diagram information to the host module for further evaluation or simulation.
3) Provides a simple mechanism/workaround to allow the users to import or export diagram information in the browser environment.
4) The client module can be scaled down to become a light-weighed diagram viewer, a diagram editor for creating diagrams in the HTML document, or a diagram simulator for demonstration purpose.
5) The client module should be flexible enough to handle new attribute in the diagram.
6) The client module should be able to load additional module on demand. For example, output graph viewer.

Server
1) The server module should be able to interact with one or more external applications, e.g. MATLAB, PSPICE, MATHEMATICA, MATHCAD, etc., if necessary.
2) It should be able to communicate and interact with multiple clients at any time.
3) It should be able to process the output data as well as to handle the errors or exceptions generated by the external applications.
4) It provides the design depository for clients to query and to save the new design examples and problems.

Data transfer format
1) It should be vendor independence and protocol independence.
2) It should be easily to be shared between applications.

To this end, a platform-independent architecture is proposed and, as an example, a trial web-based designer/simulation tool utilizing this architecture is designed and implemented in the areas of signals, systems, and control. As the result of this effort, the undergraduate engineering curriculum will be greatly improved through incorporating this innovative and effective educational tool. Students will be able to use it in such ways as visualizing key concepts, developing necessary skills for problem solving, conducting self-evaluation, and monitoring their own progress. This tool will make engineering learning become more efficient, interesting and accessible. According to our preliminary trials, students have showed a lot of interests in and benefited from using the tool to solve challenging engineering problems.
3. Proposed Architecture

3.1 General Information

The proposed architecture can be divided into two components. The first component is the client/server program modules, and the second one is the data transfer standard. Java and Extensible Markup Language (XML) have been recommended as the programming language for the development effort and as the primary data transfer and exchange mechanism. The reason Java was selected is because Java is an Internet solution. Moreover, Java is platform independent and network library ready.

XML is a markup language, which is a subset of the Standard Generalized Markup Language (SGML) and superset of the Hypertext Markup Language (HTML). It has been designed for ease of implementation and for interoperability with both SGML and HTML. XML is information design methodology especially for the web use. XML allows the designers to define their own markup tags and markup language, which will be used to encapsulate the data in the way the designers want. Moreover, XML is an ideal medium for data exchange since it can store not only the data but also the structural information. XML encode a description of the document's storage layout and logical structure and provide a mechanism to impose constraints on the storage layout and logical structure. All the information related to the diagram can be stored inside of the XML document. Comparing storing the information into propriety format that is difficult to reuse, using XML will make the server program to be less dependent to the external application.

3.2 Detail Architecture

Figure 1 shows the basic elements of the architecture. At the client side, a Java applet is running inside of a web browser. At the server side, a Java server program module is running in the web server machine. It contains the two different XML parsers and data processing modules. All diagram information (i.e., all descriptions of mathematical models) and graph data point information will be transmitted as XML documents.

A typical operation in software works as follows. First, a Java applet is loaded into the web browser. After the applet is running, user can either draw a diagram from scratch or load the diagram from the server or through import dialog window. The Java applet can be considered as stand-alone tool for a diagram designer. After the diagram and operation parameters (for the purpose of simulation or analysis) are defined, the user can initiate the applet to establish a connection with the server module. Diagram information will be compiled as a XML document and is sent over network using any network protocol (e.g. TCP/IP, Java RMI or CORBA). User also can save the diagram as XML document in the server or export the document to the web browser.

In the server module, diagram information will be re-constructed from the XML document by passing through a XML parser. If it is a "save" request, it stores the submitted XML document into the diagram depository. If it is "simulation/analysis" request, then it will be further processed to generate the calling procedures or files, which be used by the external application. After that, the server module will call the external application to execute the requested calculation or simulation on the behalf of the client applet.

After the analysis or simulation is completed, the server module will scan through the output data to check for errors or exceptions being raised. If the errors are found in the output or exceptions are raised during execution, the server module will try to handle the

![Figure 1. Architecture of Web-based simulation](image-url)
errors or quit the program gracefully and then send the error message back to the client applet. Otherwise, the output data will be processed and then written into a different kind of XML document by the second XML parser. Finally, the document will be sent back to the originated client applet. The output data will be presented in the applet in the pre-defined format. If the output data are points of the graph plot, the client applet will invoke graph plotting Java Bean to display the result. If it is "load" request, it will query diagram depository for the requested XML document and send the XML document back to the client.

4. An Implementation of the Proposed Architecture

4.1 System Overview

We have designed and implemented a prototype application using the proposed architecture in Java. It will be a browser/server-based application. In this application, there are two main program modules: client GUI and server program. The client GUI will be run inside of a Java-enabled web browser. It allows the user to dynamically construct a system block diagram in the client side GUI and then send the information to server side for the simulation. At the server-side, the server program will reside and run in the web server machine. It will listen to the requests from the client GUIs and then interface to the external engineering and simulation tool, MATLAB (chosen as a testing environment), which runs the actual simulation or conducts analysis. Then the simulation/analysis output, probably data points (for plotting graph,) will be sent back to the client for display in the graph-plotting module. Furthermore, the markup language designed for this application (used as data transfer) is called Control Block Diagram Markup Language (CBDML).

Figure 2 shows the detail information on the simulation process. The web user mainly interacts with the control system block diagram or the diagram's parameters in the client GUI. When the user initializes the simulation, the GUI will establish a TCP/IP socket connection with the server and generate the CBDML document according to the block diagram. Then it will send the document to the server program through socket. In the server-side, connection manager accepts the request from client GUI. The server program will extract the diagram information from the CBDML document and translate them into m-files through the CBDML processor. Simulation manager then executes the simulation, using recently created m-files, in the MATLAB. After the simulation is completed, the server will check for errors and read the simulation result. Finally, it will send the result back to the client GUI for displaying the result.

4.2 The Client side GUI

The client side GUI is a diagram designer, and it provides an easy-to-use interface to the user. It is the major part of the web-based block diagram simulation program. When the user opens the web page containing this GUI applet, the GUI will be downloaded from the server to the user’s web browser. When the user is drawing a control system block diagram, the designer keeps track the components inside of the design environment and maintains the diagram information in the binary
format in the local memory. Once completing the block diagram, the user can initialize the simulation from the GUI. The block diagram information will be translated into CBDML format and be transferred to the server over the network.

4.3 The Server Program

The server program is a simulation broker and provides the services to the client GUI. The server program runs from the web server and listens a specific port for request. Its main functions is to accept requests from the client GUI, process the CBDML document, prepare data and passing them to the external application (MATLAB) as well as retrieve the result from the simulation result from the external application (if the simulation/analysis is executed successfully). It consists of the connection manager, CBDML processor and simulation/analysis manager (see Figure 2).

4.4 CBDML

CBDML is an application of Extensible Markup Language (XML) 1.0, which defines a format for encoding control system block diagram together with additional markup to describe how that information may be organized, displayed, edited and used. The following is an overview of the way the proposed CBDML is organized.

The goal of this markup language is to create a new universal data format to describe a control system block diagram. It will be utilized in the above application as a data protocol and a descriptive language. On the one hand, the data file, written using this markup language, will be sent back to server for simulation/analysis purpose. On the other hand, the same file can be saved and be put into the web page. Moreover, it is vendor and application independent. However, its output should be easily shareable and useable by the different engineering applications.

The Control Block Diagram Markup Language is written in the syntax of XML and is based on the data structures of the control system block diagram from several commercial applications. CBDML is a text version of the control system data structures. It supports the markup of control system blocks and block diagrams in the same way the HTML supports the markup of textual information. In the CBDML, the content is composed of sections described different aspects of the control system diagram.

CBDML document can be divided into three major sections. The first section is the diagram information. In the diagram information section, it describes the author information (e.g. name and comment), the diagram information (e.g. title, type, version, created date, and keyword.) The second section is the system component specification section. In this section, it describes all components in the control system block diagram. There are five basic components. They are source component, block component, connector component, sink component, and connection/wire component, which corresponds a component available in the client GUI. For the first four components, all the information about a specific component will be stored within a pair of the component tags. For the connection component, in most cases, it is irrelevant to mathematical computation but it is stored for completeness and re-constructing the diagram in the future.

The third section is simulation configuration section. In this section, it describes the configuration of a simulation, such as simulation algorithm type, simulation period, and initial values.

Besides using Java applet, CBDML document can be rendered into text format using Extensible Style Language (XSL). XSL is a markup language used to specify the graphical layout, such as color, font size, and positioning of text, on the XML document. It also allows creating procedures for as reordering of information and document queries. Using XSL, the diagram information can be extracted from the CBDML document and can be display as any given format, such as table or list. For example, user may want to show only the internal functions in the diagram. Instead of using CBDML viewer, requires downloading Java applet, the user can create a XSL file to describe the ordering, query, formatting and layout of the diagram information. Currently, XSL is supported by Internet Explorer 5.

4.5 Simulation/Analysis Example

Let's consider a cart with an inverted pendulum hinged on top of it as shown in Figure 3. For simplicity, the cart and the pendulum are assumed to move in only one plane, and friction, the mass of the stick, and the gusts of wind are neglected. The control problem is to maintain the pendulum at vertical position. For instance, if the inverted pendulum is falling in the direction shown, the cart is moved to the right and exerts a force, though the hinge, to push the pendulum back to vertical position.
Using Newton’s law of pendulum and Laplace transformation, the above system can be represented as a system of state-variable equations (see Model 1). For simplicity of nonlinear models exactly the same (except that, for nonlinear systems, transfer function option should not be used). In this example, only linear analysis and simulation results are presented in the paper. The proposed architecture and CBDML handle linear and nonlinear models exactly the same (except that, for nonlinear systems, transfer function option should not be used). The same system can be expressed as block diagram shown in the figure 4. There are two source components, four block components, two connector components and a sink component. Table 1 shows all the internal functions of the components in the block diagram as state space and transfer function.

Figure 3. A cart with an inverted pendulum

Model 1 A cart equipped with an inverted pendulum

Figure 4 Block diagram of the cart with an inverted pendulum

<table>
<thead>
<tr>
<th>Source 0</th>
<th>(r = 0)</th>
<th>(r = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source 1</td>
<td>(d = 0.01 \sin \omega t)</td>
<td>(d = 0.01 \sin \omega t)</td>
</tr>
<tr>
<td>Connector 0</td>
<td>(u_1 = r - w_1 - w_2)</td>
<td>(u_1 = r - w_1 - w_2)</td>
</tr>
<tr>
<td>Connector 1</td>
<td>(u_2 = u_1 + d)</td>
<td>(u_2 = u_1 + d)</td>
</tr>
<tr>
<td>Block 0</td>
<td>(G(s) = \frac{-0.2}{s^2 + 10.78})</td>
<td>(x_1 = x_2, \ x_2 = 10.78x_1 + u_2, \ y_3 = -0.2x_1)</td>
</tr>
<tr>
<td>Block 1</td>
<td>(G(s) = \frac{9.18 - s^2}{s^2})</td>
<td>(x_1 = x_2, \ x_2 = u_4, \ u_4 = y_3 y_4 = 9.182x_1 - u_1)</td>
</tr>
<tr>
<td>Block 2</td>
<td>(G(s) = \frac{1200[1+ \frac{-83x}{s^2 + 65x + 1050}] }{})</td>
<td>(x_1 = x_2, \ x_2 = -65x_2 - 1050x_1 + y_4, \ w_1 = 1200(y_2 - 1090 x_1 + 83x_2))</td>
</tr>
<tr>
<td>Block 3</td>
<td>(G(s) = \frac{-885.53 - s}{s^2 + 19.7595})</td>
<td>(x_1 = -19.7595x_1 + y_3, \ w_2 = -885.53(y_2 - 13.7595 x_1))</td>
</tr>
</tbody>
</table>

Table 1 Functions inside the block diagram
For each component in the block diagram, the user needs to specify the attributes, such as type, frequency and amplitude for the sources, as well as time, input, output, internal state variables (or the transfer functions) for the blocks. Under current implementation, all the functions should be written as same as the way that in the MATLAB environment.

After the applet is loaded, the user first set the diagram to either state space (default) or transfer function block diagram. The user then can start to draw the block diagram in GUI workspace, figure 5. User can create a component by clicking the one of the component buttons on the toolbar and then drag-and-drop it to proper location. After that, the user needs to specify the values of the attributes of each component by double clicking on the component. A component window will be popped up. Inside the component window, user can specify the component name, attribute values and mathematical formula of the component. After specifying all the components, the user clicks simulation button from the toolbar. The simulation window will be popped up. The user set the parameters of the simulation (such as start time, stop time, and simulation algorithm being used) and finally hit "Start Simulation" button to start the simulation. The client GUI will establish a connection with the server, generate the CBDML document and send it over the network.

In the server-side, server program accepts the connection and receives the CBDML document. It parses and translates the document to m-files through CBDML processor. The server program then executes the m-files in the MATLAB. After simulation ends, it checks the result of the execution and the result (or error message) will be sent back and shown in the text area in the simulation window.

Using this application, the students can draw a simple control system block diagram and run the simulation through a web interface. The actual simulation occurs in the server. Moreover, they can evaluate the different conditions of the same system by changing the values of the attributes or parameters and observe the change in the simulation result. Furthermore, they can also add/remove component into/from the control system block diagram and evaluate the effect on the system. The same idea can be extended to such areas as circuit analysis, signal analysis, communication, etc.

5. Implementation Enhancement

The key advantage of this architecture is the high flexibility on the network, data and back-end implementations. For network implementation, the designer can choose among TCP/IP socket, Java-RMI (Remote Method Invocation) and CORBA (Common Object Request Broker Architecture). For the data implementation, the implementation designers can design their own XML-based markup language or re-use existing XML implementation as a part of the new XML implementation. For example, Mathematical Markup Language (MathML) can be used to represent the mathematical formulas (e.g. in the block component) and Vector Markup Language (VML)/Scalable Vector Graphic (SVG) can be used to represent component graphics within the designer-defined block diagram markup language, such as CBDML.

For back-end implementation, within MATLAB itself, there are a dozen of add-in MATLAB toolboxes, such as signal processing toolbox, control system toolbox, numeric analysis toolbox or ODE suite. Each toolbox provides unique functionality to the MATLAB workspace environment. By implementing this architecture, the software product can interface with MATLAB directly. Hence, it can communicate with any MATLAB toolboxes available in the workspace. This feature makes most of the functionality in the MATLAB available in the web environment. On top of that, Java can
run the applications (PSPICE, MATLAB, MATHEMATICA, etc.) from the command line through its runtime object. Java also can load other local application libraries and interface with these methods directly. The software suite can be implemented to communicate and run any application available in the web server machine.

6. Conclusion

This article describes web-based simulation/analysis architecture for the engineering education. The goal of the research is to investigate and design a simple architecture that provides flexible and web-enhanced learning environment and capabilities for engineering students to master fundamental knowledge. This architecture allows the engineering students to present, test and evaluate their own idea or simulate engineering problems (such as electrical circuit problems, control system problems, DSP problems, etc.) through a GUI inside of the web browser environment. So far, the basic architecture and the prototype for MATLAB (described in this paper) have been designed and implemented. It is expected, upon completion, a comprehensive online simulation/analysis environment compatible to different kinds of model descriptions and to typical target platforms will be available for students’ use in and outside the classroom. Once successfully proof of concept on a control system course, the implementation will be expected to be available to the other courses under the electrical and electronic engineering. Because of the high flexibility of this architecture, a new implementation (based on this architecture) can be developed for other engineering curriculum, such as civil engineering, mechanical engineering or robotic engineering.

References


Wai L. Chan

Master of Science in Computer Engineering, specializing in Software Engineering. He works as Software Engineer in Course Development & Web Services, University of Central Florida. He also is a Sun Certified Programmer for the Java™ 2 Platform. He currently involves in the university web site redesign and develops web-based applications for the distributed learning in UCF. E-mail: wlchan@mail.ucf.edu.

Zhihua Qu

Dr. Qu was born in Shanghai, China in 1963. He received his B.Sc. and M.Sc. degrees in electrical engineering from the Changsha Railway Institute in 1983 and 1986, respectively. From 1986 to 1988, he worked as a faculty member at the
Changsha Railway Institute. He received his Ph.D. degree in electrical engineering from the Georgia Institute of Technology in 1990. Since then, he has been with the Department of Electrical and Computer Engineering at the University of Central Florida. Currently, he is currently the CAE/LINK Distinguished Professor in the College of Engineering and the Director of Electrical Engineering Program. His main research interests are nonlinear control techniques, robotics, and power systems. He has published 72 refereed journal papers in these areas and is the author of two books, *Robust Control of Nonlinear Uncertain Systems* by Wiley Interscience and *Robust Tracking Control of Robotic Manipulators* by IEEE Press. He is presently serving as an Associate Editor for *Automatica* and for *International Journal of Robotics and Automation*. He is a senior member of IEEE.

**Issa Batarseh**

*Dr. Issa Batarseh, P.E.* is an Associate Professor in the School of Electrical Engineering and Computer Science at the University of Central Florida. He received the Ph.D., M.S. and B.S. degrees all in Electrical Engineering from the University of Illinois in 1983, '85 and '90, respectively. Dr. Batarseh was a visiting Assistant Professor at Purdue University from 1989 to 1990 before joining the Department of Electrical and Computer Engineering at UCF in 1991. His research interests is power electronics, focusing on dc-to-dc power supplies design, harmonic analysis, reduction and power semiconductor devices. He has two patents and one patent application pending and 28 refereed journal and 65 conference publications. Dr. Batarseh research work has been sponsored by Florida Space Grand Consortium, Allied Signals Corp., and the Industry Application Society of IEEE. He is a registered professional engineer in Florida and a Senior Member of IEEE. Currently serving a co-advisor for IEEE student Chapter.