*Int. J. Engng Ed.* Vol. 21, No. 2, pp. 288–296, 2005
Printed in Great Britain.

0949-149X/91 $3.00+0.00


# Using XML/Java to Enhance an Online Learning Architecture for Engineering Education*

WAI L. CHAN and ZHIHUA QU
*University of Central Florida, Orlando, FL 32816, USA. E-mail: qu@mail.ucf.edu*

*This paper will present the architecture for designing and developing a web-based teaching enhancement tool for engineering education. This architecture will enhance student learning by providing an innovative way for them to interact with standard engineering software through the web. The new architecture will provide a flexible online learning environment that will allow the students to present, test and evaluate their own ideas. To demonstrate its capabilities, we will present a Java application platform using this architecture. Moreover, we shall briefly discuss an XML-based markup language, Control Block Diagram Markup Language (CBDML), that was written during the development of this application. The design of the architecture described in this article has many possible applications, but this paper only presents the core results; more applications could enrich the architecture and CBDML, thus making possible a wide variety of production ideas.*

## INTRODUCTION

IN THE PAST YEAR, several new technologies have been utilized in a number of university courses to enhance the course material and students' learning experience within traditional face-to-face classes. This practice has proved to be very effective for many engineering students. According to several independent studies, many engineering freshmen and sophomores have difficulty grasping the basic concepts being taught in introductory engineering courses. The reason often is that they cannot visualize the concepts being taught. Moreover, engineering students may not be equipped with the necessary skills to produce creative designs using critical thinking processes [1]. Those students who cannot understand the fundamental principles of engineering often fail in engineering courses and even drop out.

Over the past few years, several nationwide initiatives have developed multimedia and web-based educational tools [2–4]. The majority of such efforts have focused primarily on the graphical user interface (GUI) for the purposes of static display/presentation. Several initiatives have focused on developing web-based simulation tools, but only targeted to a specific design or operation [4]. None of these has attempted to design and develop a complete set of solutions for online learning.

On the other hand, the existing tools that are often used in engineering courses are standalone applications and are usually installed on the computers in the laboratory/computer center. This means that the user needs to be in front of the computer in order to use them. If the students are off-campus, they may not have the opportunity to get practice with the tools. In addition to this, more and more course work is assigned to be completed using a computer which has such engineering software installed. This can mean that the computer laboratory becomes overcrowded. Furthermore, these tools have mostly been developed for computational instead of educational use. Although they can be used for educational purposes, basic training is usually necessary so that students are able to fully utilize the tools.

The architecture discussed in this paper is part of the engineering education enhancement project that will change the way we teach engineering courses to freshmen at the University of Central Florida. Through this project, a team of faculty members, computer engineering graduate students and software engineers will develop fully web-enhanced software packages. These will allow engineering students to design, build and simulate engineering problems (such as electrical circuit problems, control system problems, DSP problems, etc.) through a GUI inside the web browser. This will provide an innovative method for interactive teaching and online design testing. The online computer simulation of engineering problems will help students to better understand abstract engineering principles and concepts, and will offer multiple opportunities to solve engineering design problems [5–7]. In addition, since these packages will be available through the web, it will not be necessary to install anything on the client side. The students can simply have access to this

* Accepted 25 March 2004.

powerful software at home or in the computer laboratory. The software maintenance and upgrading are also hassle-free [5–7].

## OBJECTIVES

The main goal of this research is to investigate and design a simple architecture that provides a flexible learning environment and the option for engineering students to either work through a set of problems posted by the instructor or present, test and evaluate their own ideas through the web. This architecture should enhance the student's ability to understand basic concepts and principles through examples, scenarios and exercises.

On the client side, the architecture provides a workbench-like 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. The client should also be able to send a request and diagram to the host module for further evaluation or simulation. The architecture should provide a simple mechanism/workaround to allow users to import or export diagram information in the browser environment. The client module should also be scalable. It can be scaled down to become a light-weight diagram viewer applet, a diagram editor applet for creating diagrams within the browser, or a diagram simulator applet for simulation/demonstration purposes. It should be flexible enough to handle new attributes in the diagram as well as to load additional modules on demand. For example, an output graph viewer can be used.

On the server side, the server module should, if necessary, be able to interact with one or more external applications, e.g. MATLAB, PSPICE, MATHEMATICA, MATHCAD, etc. It should be a multi-thread server capable of interacting with multiple clients at any one time. Furthermore, it should be able to process output data as well as handle the errors or exceptions generated by the external applications. If the system has access to a relational database, it should also provide a design depository where clients can post any queries and save design examples and problems. Finally, the data transfer format also has an important role in this architecture. It should be vendor independent and protocol independent. It should be easily shared between applications and convert into any proprietary application format.

To this end, a platform-independent architecture is proposed and, as an example, a trial web-based designer/simulation tool utilizing this architecture has been designed and implemented in the areas of signals, systems, and control. The undergraduate engineering curriculum will be greatly improved by incorporating this innovative and effective educational tool. Students will be able to use it to visualize key concepts, develop necessary skills for problem-solving, conduct self-evaluation, and monitor their own progress. This tool will make engineering learning more efficient, interesting and accessible. According to preliminary trials, students have shown a lot of interest in the tool and have benefited from using it to solve challenging engineering problems.

## PROPOSED ARCHITECTURE

*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 used as the programming language as the primary data transfer and exchange mechanism [8]. 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 a 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 an information design methodology that is designed for use with the web. 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 encodes a description of the document's storage layout and logical structure and provides a mechanism for imposing constraints on the storage layout and logical structure. All the information related to the diagram can be stored in the XML document. Compared with storing information in a proprietary format that is difficult to re-use, using XML will make the server program less dependent on the external application.

*Details of the architecture*

Figure 1 shows the basic elements of the architecture. On the client side, a Java applet is running inside a web browser. On 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, the user can either draw a diagram from scratch or load a diagram from the server or through the import dialog window. The Java applet can be considered as a standalone tool for a diagram designer. After the diagram and operation parameters (for the purpose of simulation or analysis) are defined, the user can activate the applet to

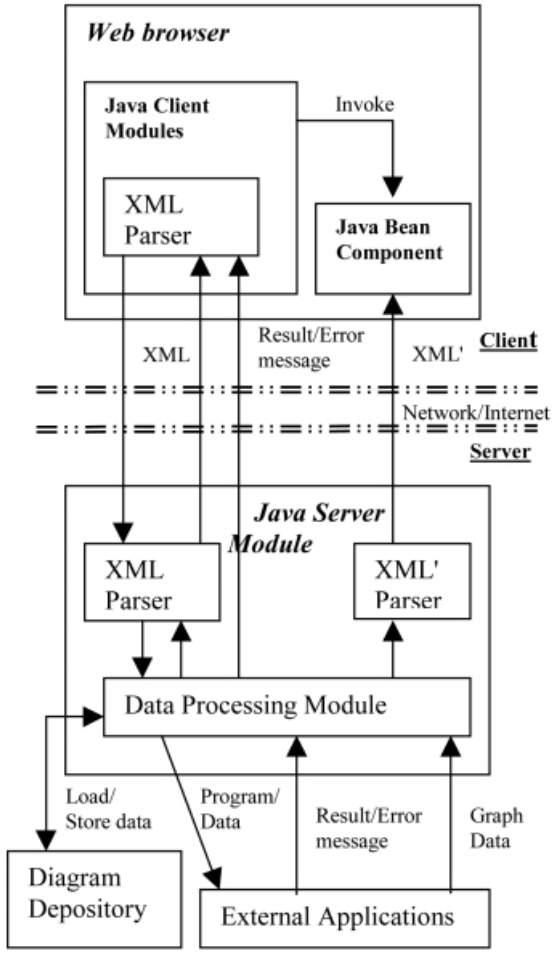


Fig. 1. Architecture of the web-based simulation.

establish a connection with the server module. Diagram information will be compiled as an XML document and this is sent over the network using any network protocol (e.g. TCP/IP, Java RMI or CORBA). The user can also save the diagram as an XML document on the server or export the document to the web browser.

In the server module, diagram information will be reconstructed from the XML document by passing it through an XML parser. If it is a 'save' request, the submitted XML document is stored in the diagram depository. If it is 'simulation/analysis' request, it will be further processed to generate the calling procedures or files which will be used by the external application. After that, the server module will request the external application to execute the desired calculation or simulation on 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 and see if exceptions occur. If errors are found in the output or exceptions arise during execution, the server module will try to tackle the errors or will quit the program gracefully and send an 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 originating client applet. The output data will be presented in the applet in a predetermined format. If the output data are points of a graph plot, the client applet will invoke the graph-plotting Java Bean to display the result. If it is a 'load' request, it will ask the diagram depository for the requested XML document and send it back to the client.

## AN IMPLEMENTATION OF THE PROPOSED ARCHITECTURE

*System overview*

We have designed and implemented an application platform using the proposed architecture in Java. This will be a browser/server-based application. In this application, there are two main program modules: the client GUI and the server program. The client GUI will run in a Java-enabled web browser. This will allow the user to dynamically construct a system block diagram on the client GUI and then send the information to the server side for the simulation. The server program will run in the web server machine. It will attend to the requests from the client GUIs

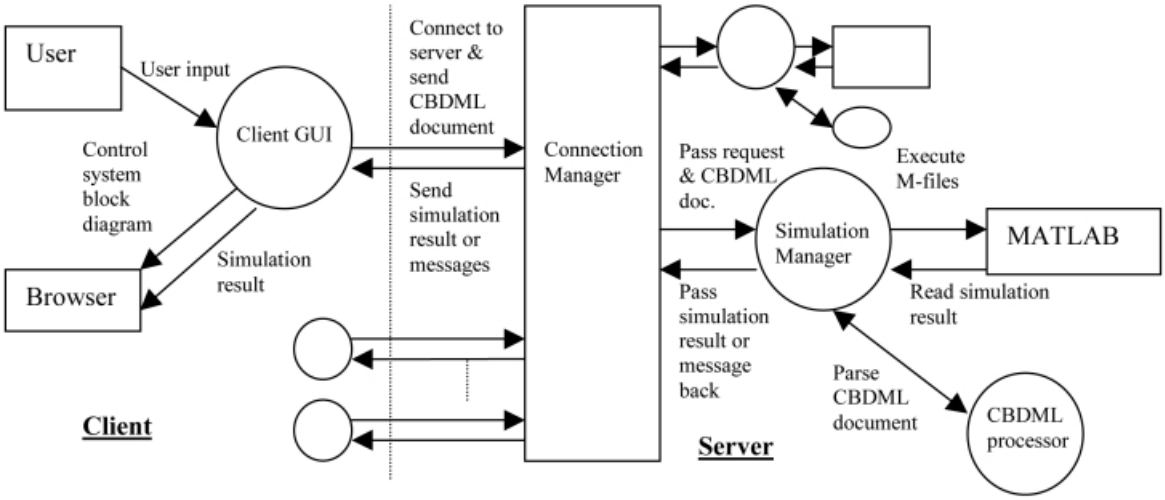


Fig. 2. The simulation process of the Java application.

and then interface with the external engineering and simulation tool, MATLAB [9] (chosen as a testing environment), which runs the actual simulation or conducts analysis. Then the simulation/analysis output, probably data points (for plotting graphs), will be sent back to the client for display in the graph-plotting module. The markup language designed for this application (used as data transfer) is called Control Block Diagram Markup Language (CBDML).

Figure 2 shows 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 the socket. On the server side, connection manager accepts the request from the client GUI. The server program will extract the diagram information from the CBDML document and translate it into m-files through the CBDML processor. The simulation manager then executes the simulation, using recently created m-files, in 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.

*The client GUI*

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

*The server program*

The server program is a simulation broker and provides services to the client GUI. The server program is run from the web server and receives requests. Its main function is to accept requests from the client GUI, process the CBDML document, prepare data and pass them to the external application (MATLAB), as well as retrieve the result from the simulation result on the external application (if the simulation/analysis is executed successfully). It consists of a connection manager, a CBDML processor and a simulation/analysis manager (see Fig. 2).

*CBDML*

Control Block Diagram Markup Language (CBDML) is an application of Extensible Markup Language (XML) 1.0, which defines a format for encoding control system block diagrams 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 the server for simulation/analysis purposes. On the other hand, the same file can be saved and be put on the web page. Moreover, it is vendor and application independent. However, its output should be easily shared and used 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 structure. It supports the markup of control system blocks and block diagrams in the same way that HTML supports the markup of textual information. In CBDML, the content is composed of sections describing different aspects of the control system diagram.

CBDML documents can be divided into three major sections. The first section is the diagram information. In this section, the author information (e.g. name and comment) and diagram information (e.g. title, type, version, date created, and keyword) are described. The second section is the system component specification section. It describes all components in the control system block diagram. There are five basic components. They are the source component, block component, connector component, sink component, and connection/wire component, which corresponds to a component available in the client GUI. For the first four components, all the information about each will be stored in a pair of the component tags. In most cases, the connection component is not relevant to mathematical computation but it is stored for completeness and for reconstructing the diagram. The third section is a simulation configuration section. It describes the

```
<BlockDiagram size="800 600" type="SS">
    <diagram.info>
        <title>Testing Block Diagram</title>
        <author>Wai Chan</author>
        <version>1.0</version>
        <comment keywords="test comment">
            This is a test comment</comment>
    </diagram.info>

    <diagram.block>
        <component.source type="sin" id="s0" id.to="c0" size="80 60"
            location="10 200">
            <component.name>Source</component.name>
            <source_type>Sine Wave</source_type>
            <freq>10</freq>
            <freq_range>0</freq_range>
            <amp>5</amp>
            <amp_range>0</amp_range>
        </component.source>
        <component.block time="t" input="u1" output="y1" state="x1"
            id="b0" id.from="c0 " id.to="b1 c0 " size="100 60"
            location="320 200">
            <component.name>Block A</component.name>
            <function> .... </function>
            ....
        </component.block>
        <component.block time="t" input="u2" output="y2"
            state="x1 x2" id="b1" id.from="b0" id.to="i1 c0"
            size="100 60"  location="500 200">
            <component.name>Block B</component.name>
            <function> ... </function>
            ....
        </component.block>
        <component.connector id="c0" id.from="s0 b1 b0 " id.to="b0"
            size="60 60" location="160 200">
            <component.name>Connector</component.name>
        </component.connector>
        <component.sink id="i0" id.from="b1" size="80 60"
            location="710 200">
            <component.name>Scope</component.name>
            <horiz_range>20</horiz_range>
            <vert_range>2</vert_range>
        </component.sink>
        <component.connection id="w0" size="113 3" location="599 240"/>
        <component.connection id="w1" size="83 3" location="419 240"/>
        ....
        <component.connection id="w9" size="3 64" location="178 139"/>
    </diagram.block>

    <diagram.control type="rk3" start="0.0" stop="2.0" max_step="0.01"
        min_step="0.001">
        <simulation_type>Runge-Kutta 3</simulation_type>
    </diagram.control>
</BlockDiagram>
```

Script 1. CBDML document example.

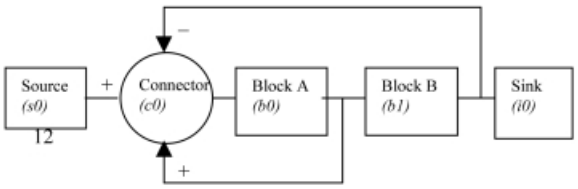


Fig. 3. Control system block diagram example.

configuration of a simulation, such as the simulation algorithm type, simulation period, and initial values.

*CBDML example*

Script 1 describes the block diagram shown in Fig. 3. In the diagram, there are two block components and one connector component. For block A, the input is from the connector and the output is to block B. However, the same output also feeds back to the connector. For block B, the input is from block A and the output to another component. Also, the same output feeds back to the connector. The connector has three inputs (from source, block A and block B) and one output (to block A).

The first line of the CBDML document defines the type and size of the block diagram workspace as state space (SS) and as $800 \times 600$. The next eight lines of the CBDML document are a section on diagram information. This defines the non-technical attributes of a diagram, such as title, author, version, keyword and author's comment.

The next 30 lines comprise the diagram specification section. This section can be further divided into six sub-sections. The first sub-section is the specification of the source component. It specifies the attributes of a source component, such as the name, type, size and location of the component, as well as the frequency (10Hz) and amplitude (5 units) of the source. The second and third sub-sections specify the block components (there are two block components in Fig. 3). The name, type, size and location of the component, local variables, connection information (e.g. connection from connector c0 to block b0) and the block's numerical

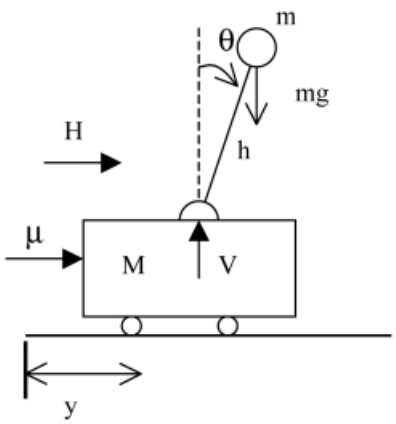


Fig. 4. A cart with an inverted pendulum.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{-2mg}{2M+m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \dfrac{2g(M+m)}{(2M+m)h} & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{2}{2M+m} \\ 0 \\ \dfrac{-1}{(2M+m)h} \end{bmatrix} u$$

Model 1. A cart equipped with an inverted pendulum.

functions are described. In this case, the function would be expressed as MATLAB functions.

The fourth sub-section gives the specifications of the connector component. It defines the name, type, size and location of the component, and includes local variable and connection information. Currently, the connector component is assigned to the 'multiple input add junction' block. The fifth sub-section gives the specifications of the sink component, including the name and type, size and location of the component, connection information and display parameters. The sixth sub-section specifies the connection/wire components. It describes the orientation, length and location of the component. The remaining four lines comprise a section on the diagram simulation configuration. This defines the configuration parameters of the simulation, such as the type (rk3) and name (Runge-Kutta 3) of the simulation, the simulation period (0.0–2.0 seconds) and sampling rates (maximum step: 0.01, minimum step: 0.001).

Besides using the Java applet, the CBDML document can be rendered into text format using Extensible Style Language (XSL). XSL is a markup language used to specify the graphical elements, such as color, font size, and positioning of text in the XML document. It also permits procedures such as reordering of information and document queries. Using XSL, the diagram information can be extracted from the CBDML document and displayed in any given format, such as a table or a list. For example, users may want to show just the internal functions in the diagram. Instead of using a CBDML viewer, which necessitates downloading the Java applet, the user can create an XSL file to define the ordering, queries, formatting and layout of the diagram information.

*Simulation/analysis example*

We will now consider a typical control system problem: a cart with an inverted pendulum hinged on top of it, as shown in Fig. 4. For simplicity, the cart and the pendulum are assumed to move in only one plane, and friction, the mass of the stick, and gusts of wind are ignored. The goal is to maintain the pendulum in the 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, through the hinge, to push the pendulum back to the 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

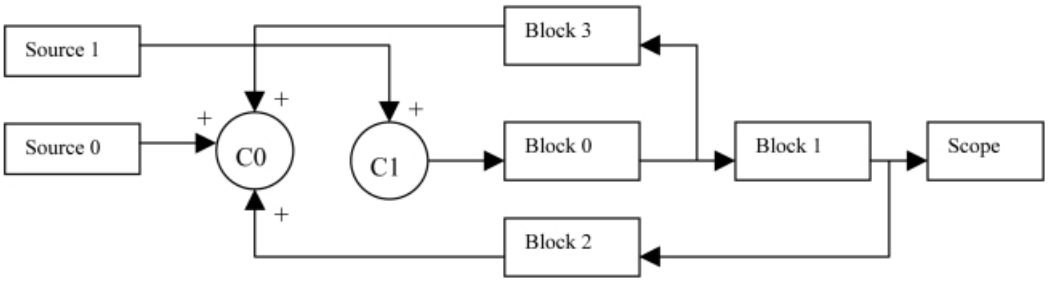


Fig. 5. Block diagram of the cart with an inverted pendulum.

Model 1). In this example, only linear analysis and simulation results are presented. The proposed architecture and CBDML handle linear and nonlinear models exactly the same (except that, for nonlinear systems, the transfer function option should not be used). The same system can be expressed as a block diagram, shown in Fig. 5. 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.

For each component in the block diagram, the user will 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 the same way as those in the MATLAB environment.

After the applet is loaded, the user first sets the diagram to either state space (default) or transfer function block diagram. The user then can start to draw the block diagram in the GUI workspace (see Fig. 6). Users can create a component by clicking one of the component buttons on the toolbar and then dragging it to the required 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 pop up and in this the user can specify the component name, attribute values and the mathematical formula of the component. After specifying all the components, the user clicks on the simulation

Table 1. Functions in the block diagram

| | Transfer Functions | State Space Function |
|---|---|---|
| Source 0 | $r = 0$ | $r = 0$ |
| Source 1 | $d = 0.01 \sin \omega t$ | $d = 0.01 \sin \omega t$ |
| Connector 0 | $u_1 = r - w_1 - w_2$ | $u_1 = r - w_1 - w_2$ |
| Connector 1 | $u_2 = u_1 + d$ | $u_2 = u_1 + d$ |
| Block 0 | $G(s) = \frac{-0.2}{s^2 + 10.78}$ | $x_1 = x_2,\ x_2 = 10.78x_1 + u_2,\ y_3 = -0.2x_1$ |
| Block 1 | $G(s) = \frac{9.18 - s^2}{s^2}$ | $\dot{x}_1 = x_2,\ x_2 = u_4,\ u_4 = y_3 y_4 = 9.182x_1 - u_4$ |
| Block 2 | $G(s) = 1200\left[1 + \frac{-83s - 1090}{s^2 + 65s + 1050}\right]$ | $\dot{x}_1 = x_2,\ x_2 = -65x_2 - 1050x_1 + y_4,$ $w_1 = 1200(y_4 - 1090x_1 + 83x_2)$ |
| Block 3 | $G(s) = -885.53\frac{s + 6}{s + 19.7595}$ | $x_1 = -19.7595x_1 + y_3,\ w_2 = -885.53(y_3 - 13.7595x_1)$ |

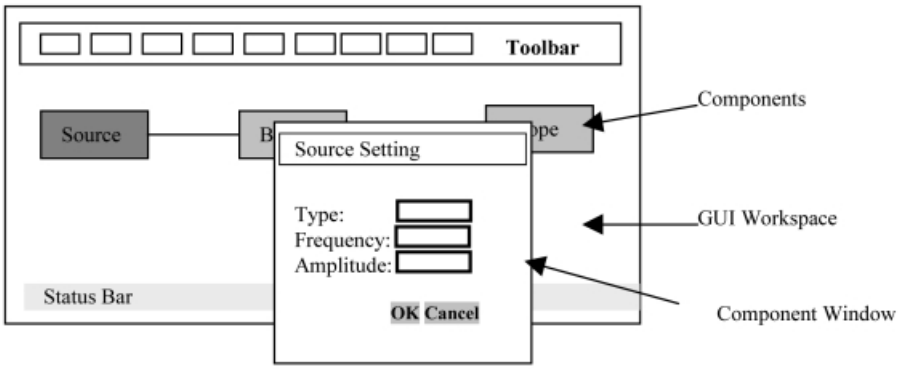


Fig. 6. The layout of the client GUI.

button on the toolbar. The simulation window will pop up and here the user sets the parameters of the simulation (such as start time, stop time, and the simulation algorithm being used), then the user hits the '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.

On the server side, the server program accepts the connection and receives the CBDML document. It parses and translates the document to m-files through the CBDML processor. The server program then executes the m-files in MATLAB. After the simulation ends, it checks the result and this (or the 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 block diagram 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 observing the change in the simulation result. Furthermore, they can also add/remove the 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.

## 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 standard TCP/IP socket communication, HTTP communication through Java-Servlet, CORBA (Common Object Request Broker Architecture) or Java-RMI (Remote Method Invocation). 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/functions (e.g. in the block component) and Scalable Vector Graphic (SVG)/Vector Markup Language (VML) 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 add-in MATLAB toolboxes, such as a 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 of 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 run-time 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.

## CONCLUSION

This article describes a web-based simulation/analysis architecture for engineering education. The objective was to investigate and design a simple architecture that provides a flexible web-enhanced learning environment and the possibility for engineering students to master fundamental knowledge. This architecture allows engineering students to present, test, and implement their own ideas or simulate engineering problems (such as electrical circuit problems, control system problems, DSP problems, etc.) through a GUI in the web browser environment. So far, the architecture and the platform for MATLAB have been designed and implemented. A comprehensive online simulation/analysis environment compatible with different kinds of model descriptions and typical target platforms will be available for students to use in and outside the classroom. This tool has been installed and configured in a departmental web server and students enrolled in control system courses will be able to take advantage of it. Reactions from the students have been very positive. Also of some importance is that the proposed platform can be used in other interactive and innovative teaching and learning settings. This means that more teaching tools can be integrated into more courses in electrical and computer engineering. Because of the high flexibility of this architecture, a new implementation (based on this architecture) could be developed for other engineering curriculums, such as civil engineering, aerospace engineering, and mechanical engineering.

## REFERENCES

1. J. B. Schodorf, M. A. Yoder, J. H. McClellan and R. W. Schafer, Using multimedia to teach the theory of digital multimedia signals, *IEEE Transactions on Education*, **39**(3) (1996), pp. 336–341.
2. C. Graham and T. Trick, Java applet enhanced learning in a freshman ECE course, *Journal of Engineering Education*, **87**(4) (1998), pp. 391–397.
3. D. Wallace and S. Weiner, How might classroon time be used given WWW-based lectures?, *Journal of Engineering Education*, **87**(3) (1998), pp. 237–248.

4. T. Webster and K. Dee, Supplemental instruction integrated into an introductory engineering course, *Journal of Engineering Education*, **87**(4) (1998), pp. 377–384.
5. W. L. Chan, Z. Qu and I. Batarseh, Web-based simulation architecture for engineering education using Java/XML, ASEE SES Annual Meeting, Virginia Tech., TA3–2, April 2000.
6. I. Batarseh, Q. Zhang, R. Eaglin, Z. Qu and P. Wahid, Multimedia enhancement of the electrical engineering core course, 2000 ASEE Annual Conference, session no. 3232, St. Louis, MO, June 2000.
7. W. Chan, The development of a web-based designer for simulating a dynamic system by remotely accessing MATLAB using Java and XML, M.Sc. thesis, Dept. of Electrical and Computer Engineering, University of Central Florida, Summer 1999.
8. D. Megginson, *Structuring XML Documents*, Prentice-Hall, Englewood Cliffs, NJ (1998).
9. A. Cavallo, R. Setola and F. Vasca, *Using MATLAB SIMULINK and control system toolbox*, Prentice-Hall PTR, Englewood Cliffs, NJ (1996).

**Wai L. Chan** works as a software engineer and team leader in Course Development & Web Services, University of Central Florida. He also is a Sun Certified Programmer for the Java™ 2 platform, Oracle Certified Java™ Developer, and IBM Certified Developer for XML. He is currently involved in the redesign of the university's main website, in the university portal and several web-based development projects for distributed learning in UCF.

**Zhihua Qu** received his Ph.D. in electrical engineering from the Georgia Institute of Technology in 1990. Since then, he has been with the School of Electrical and Computer Science at the University of Central Florida. He is currently Director of Electrical Engineering and a professor. His main research interests are nonlinear control techniques, robotics, and power systems. He has published 99 refereed journal papers and 160 conference articles in these areas and is the author of two books, *Robust Control of Nonlinear Uncertain Systems* (Wiley Interscience) and *Robust Tracking Control of Robotic Manipulators* (IEEE Press). He is presently serving as an Associate Editor for *Automatica* and for the *International Journal of Robotics and Automation*. He is a senior member of the IEEE.