A WEB-ACCESSIBLE HEAT EXCHANGER EXPERIMENT

Authors:

Clark K. Colton, Dept. of Chem. Eng., M.I.T., Cambridge, MA 02139 ckcolton@mit.edu
Marc Knight, Dept. of Chem. Eng., M.I.T., Cambridge, MA 02139 toolk@alum.mit.edu
Rubaiyat A. Khan, Dept. of Chem. Eng., M.I.T., Cambridge, MA 02139 rakhan@alum.mit.edu
Sarah Ibrahim, Dept. of Chem. Eng., M.I.T., Cambridge, MA 02139 sarah_i@mit.edu
Richard West, Dept. of Chem. Eng., M.I.T., Cambridge, MA 02139 rwest@alum.mit.edu

Abstract – *We have developed a remotely controlled heat exchanger experiment as part of the I-Lab project at MIT. It has been used on the Internet by students in a variety of chemical engineering courses, and the response of students to the online experiment has been assessed. The hardware consists of a general service unit, custom designed and fabricated for us (HT-30XC, Armfield, Ltd., England), on which is mounted a heat exchanger (flat plate, shell and tube, or double pipe). Cold water flows through the exchanger in single pass, whereas hot water is recycled to a heated reservoir. Both fluid flowrates, their direction (cocurrent or countercurrent), and the inlet hot temperature are monitored and are under the control of the students. All other inlet and outlet temperatures are monitored. Monitoring and control is carried out with a computer/web server using LabVIEW 6.1 software (National Instruments, USA). Data is published to web-accessible LabVIEW graphical user interfaces (GUI) or via a Data Socket Server (National Instruments) to a Java2 GUI. A database (Microsoft SQL) is used for registering, authentication, and scheduling (ASP.NET) and for collaboration management software (Java2), which provides for chat capabilities and ability to pass local control between team members who are collaborating on carrying out the experiment from their own computers in different locations. The experiment has been used successfully in various ways in courses. Student response has been favorable in general, and students appreciate the ability to take data from real equipment, especially in engineering science courses that otherwise contain no laboratory component. Student response improves with increasing sophistication of the GUI and is very high when everything works as intended. Conversely, student response can be very negative when problems in control or operation of the website occur. These findings suggest that web-accessible, remotely controlled laboratory experiments can be valuable in a variety of settings and that the value of the experience, as perceived by the students, depends strongly upon the quality, richness, and reliability of the website and GUI and, in the case of complex experiments, on the ability of the students to gain control of the independent variables at their disposal.*

INTRODUCTION

Web-accessible laboratory experiments are gaining popularity because they provide remote access to measurements with real equipment while making very efficient use of resources. We have

developed a remotely controlled heat exchanger experiment as part of the I-Lab project at MIT. It has been used on the Internet by students in a variety of chemical engineering courses, and the response of students to the online experiment has been assessed. In this paper, we summarize our experiment with the web-accessible remotely controlled heat exchanger experiment. Additional details are available elsewhere [1,2].

HEAT EXCHANGER SYSTEM OVERVIEW

We have worked with Armfield Ltd. (Hampshire, England) to develop a remotely controlled experiment. Armfield markets a range of small scale heat exchangers that represent the common types found in industry for indirect transfer of heat from one fluid stream to another. We acquired three heat exchangers, double-pipe or tubular (HT31), flat plate (HT32) and shell-and-tube (HT33) and one service unit (HT30XC), which was modified by Armfield for remote control by a computer.

The service unit (HT30XC) is a bench-top apparatus on which one of the heat exchangers may be individually mounted. The Service Unit provides the necessary services and measurement facilities for investigation and comparison of the different heat exchanger working principles and operating characteristics. These services include providing streams of hot water (heating fluid) and cold water (process fluid) at variable flowrates to the heat exchanger under evaluation. The service unit can also control flow direction by operating in either cocurrent or countercurrent mode. The ability to change the type of exchanger quickly, without the use of tools, and the fast response of the system under optimal control to changes in water flow rate or temperature allow the steady-state experiments to be carried out in a relatively short period of time.

The HT30XC is connected to the computer through a universal serial bus (USB) port and comes equipped with a software driver for the Windows operating system. It is designed to operate from a Windows computer, and all the parameters are computer controlled with no manual intervention other than setting the equipment up and switching it on. The software that was bundled with the HT30XC was not designed to be web-accessible, so we designed our own software architecture to put the experiment on the web.

Figure 1 is a schematic diagram of the heat exchanger flow circuit on the service unit and its interaction with computer software for monitoring and control. The single pass cold water stream is controlled by a pressure regulator, and the recirculated hot water stream is controlled by a bi-directional gear pump that allows the exchanger to be configured for cocurrent or countercurrent operation. Four k-type thermocouples monitor the inlet and outlet temperatures of the hot and cold water streams. There are also two flowmeters that measure the flowrates of the hot and cold water streams. The thermocouples, flowmeters, pressure regulator, and pump all interface with the computer via the software driver. Additionally, feedback mechanisms use input from the thermocouples and flowmeters to control the flowrates and the inlet hot water temperature by heating in the hot water reservoir.

An important aspect of the I-Lab Heat Exchanger is its real-time feature. A change in one of the flowrate values or the hot water setpoint leads to a transient response on the data output, and the desired steady state value should to be reached as soon as possible. In order to achieve this goal, control algorithms had to be implemented for the hot and cold water flowrates and the hot

water temperature. The hot water temperature was controlled by placing algorithms that determined the on/off operation of the heater, and the cold and hot water flowrates were controlled through algorithms that determined pump and valve settings. All the control algorithms were based upon proportional-integral-derivative (PID) control. Once the control algorithms were in place, the Ziegler-Nichols closed-loop tuning method was used to find the optimal PID parameters. The optimal parameters so obtained for the cold and hot water flowrates were applicable over the entire useful range (cold 0.5 to 5.0 L/min, hot 0.5 to 5.0 L/min). However, the optimal parameters for the hot water temperature control varied as a function of the hot water flowrate. The optimal parameters are very sensitive to the physical characteristics of the heat exchanger including tube lengths and diameter. Any change to the arrangement of the exchanger and tubing will render the PID parameters sub-optimal. The optimal PID parameters were calculated for each of the heat exchangers (shell and tube, flat plate, and double pipe); in addition, a different set of parameters was obtained for each mode of operation (countercurrent and cocurrent).

SOFTWARE ARCHITECTURE

Figure 2 depicts the software architecture used for the I-Lab Heat Exchanger Project. At the core of the architecture is the LabVIEW 6.1 software that communicates with the heat exchanger and makes data accessible to users of the system. The architecture also performs user registration, authentication, and scheduling and is flexible enough to allow students to work in teams on an experiment.

The HT30XC ships with software that allow the USB interface to be controlled from a computer. Included with this software is a Dynamic Link Library file (DLL), which provides a means to access the input and output parameters of the HT30XC Heat Exchanger Service Unit from any Windows program. Performing the appropriate function call to the DLL accesses the I/O data.

To perform the function calls to the DLL file, originally we used LabVIEW 6i, later we used LabVIEW 6.1, a graphical programming language developed by National Instruments that is easy to use and reduces system development time. In LabVIEW, block diagrams or VIs (Virtual Instruments) are used to develop a Graphical User Interface (GUI) to monitor and control parameters. This programming language contains a very large library of Graphical Instrument control tools, such as knobs, dials, and charts, which make creating user GUIs very easy. It also has built in functions that allow for easy use of the DLL file for the HT30XC service unit. In LabVIEW, this is accomplished via the Call Library Function, which can provide access to a particular function in a specified DLL. The Call Library Function block diagram is shown in Figure 3. Through the use of this function and the DLL file, the LabVIEW software is able to read from and write to both the analog and digital channels of the HT30XC service unit. Analog channels are used for transmitting temperature and flowrate data whereas digital channels are used to turn on and enable the service unit.

The I-Lab Heat Exchanger Project was designed to have two types of web-accessible interfaces, one type would be for instructors only, and the other type would be for students. To facilitate remote instructor control, a VNC server was used, a product from AT&T Laboratories,

Cambridge, England. The VNC server allowed the instructor to remotely view the desktop of the server computer, essentially allowing the instructor to have the same control as if he/she were seated at the server. For remote student control, two types of interfaces were developed. With LabVIEW 6i, one interface was developed using the Java language. The second interface was developed in LabVIEW 6.1, which permits remote control of the GUI on the web, when it became available. Built into the Java interface were collaboration features, whereas the LabVIEW 6.1 interface was viewed from a webpage that contained a Java Chat window in addition to the interface.

JAVA CLIENT INTERFACE

In implementing the Java interface, a DataSocket server was used to serve as an intermediary between the Java interface and the underlying LabVIEW software. The DataSocket server is a software product developed by National Instruments that enables real-time sharing of data among a variety of software clients developed in different programming languages.

The DataSocket server uses a publish-subscribe model to share the data. The server publishes real-time data to the Internet at a specific address via a dstp protocol (Data Sockets Transfer Protocol). Each data point binds to a specific URI (with a prefix dstp://). In our case, the LabVIEW GUI sends data to the URIs, and the client Java interfaces subscribe to the server using these specific data addresses and listen for updates. On the other hand, a Java interface can also publish parameter control values to the DataSocket server while the LabVIEW software listens for updates. Both of these situations occur in order to provide proper control via the Java Interface.

The Java interface is divided into many sections as depicted in Figure 4. The login panel handles user authentication, and relevant data can be viewed on the chart or in the data output panel. All the relevant experiment parameters are contained in the parameter controls panel, and the collaboration window panel contains all elements of the system's collaboration capability.

In the login panel, a user must enter his/her Login ID and Password and press the "Login" button. Every user is assigned a Login ID and Password when they first register to use the system. Requiring this information provides an additional level of authentication to ensure that the user is authorized to perform the experiment. When the "Login" button is pressed, the application creates a socket connection with the collaboration server and passes to it a Login message. Once a user is logged in, he/she will be able to view temperatures and flowrates, and their Login ID will appear in the collaboration panel.

Students can change the experiment parameters by submitting values via the parameter controls panel. In the parameter controls panel, the student may have control over the:

- Flow direction (Countercurrent or Cocurrent)
- Hot fluid flowrate
- Cold fluid flowrate
- Temperature setpoint
- PID (Proportional-Integral-Derivative) parameters

Students must click the Submit button to update the parameter values. Depending on the assignment, students may not have control over all the parameters. If a student does not have control of the parameter, the corresponding button will appear gray.

Shown in the data output panel are the actual temperature and flow values. The temperature values are received from thermocouples mounted at different locations in the heat exchanger. For the flat plate HT32 and the shell-and-tube HT33 heat exchanger, the thermocouple locations are:

- Hot fluid inlet (T1)
- Hot fluid outlet (T2)
- Cold fluid inlet (T3)
- Cold fluid outlet (T4)

For the double pipe HT31 heat exchanger, two additional thermocouples are located at the midpoint between two sections (Hot fluid midpoint, T5; Cold fluid midpoint, T6).

The chart panel contains a scrolling graph that can display these four temperatures. Users can select which combination of temperatures they wish to view by selecting the appropriate checkboxes on the side.

The data recording panel allows students to save their data in Excel format in a folder on the server. There is a text box to input the filename and a button that submits the filename to the server. The student can press the start recording button only after the submit button had been pressed. A green light will appear when recording starts after which he/she can press the Stop Recording button to stop the data recording.

The collaboration panel is divided into two distinct components. Starting at the top is the chat component. On the left is a list of all users currently logged in and on the right is the chat window, where users can view messages. Below these two boxes is a field where users can enter messages and send them to other users.

Below this field is the "Pass the Token" component that controls which user can control the system parameters shown on the left hand side of the interface. If a user does not have control over a parameter, the Submit button next to it will appear gray. The name of the user that currently has control over the parameters appears next to the label "Person with control." The first user to log into the system automatically gets control of the parameters and his/her Login ID will appear in this box. If a user wants to gain control of the parameters, he/she must press the button labelled "Request for Control", which broadcasts a message to all users who have logged in via the chat window. The user who has control of the parameters can transfer control to the user who made the request by typing the user's username into the "Pass Control" field and pressing the appropriate button.

On the server side, there is a collaboration server running which enables these features. Its basic job is to take a message from one client interface, interpret it, and broadcast the appropriate message to all the other clients connected to the server. For example, if a student enters a message for the chat window, it is sent to the server that then broadcasts it to all the other clients, and the message appears on all the other interfaces. Additionally, the server can broadcast messages about who should have control of the equipment, which would enable/disable control over the parameters.

LABVIEW CLIENT INTERFACE

LabVIEW 6.1 was the first version of LabVIEW to offer web-based control of a LabVIEW VI. As a result, we decided to develop a client GUI using LabVIEW 6.1 for two main reasons:

1. To explore the possibility of reducing system development time, as compared to Java programming, if additional pieces of equipment were to be made web-accessible.
2. To provide students with a more user-friendly interface which more closely resembles a control panel that they might find on a laboratory instrument.

It was our hypothesis that students would prefer to use the LabVIEW interface since the controls included knobs, buttons, and real-time, scaleable scrolling graphs. Figure 5 depicts the LabVIEW Client Interface that was developed. This GUI is embedded into a webpage that also contains a Java chat window.

The GUI is divided into two pages. On the page shown in Figure 5, there are controls and indicators divided into three distinct sections: Parameter Controls, Temperature Data, and Flow Rate Data. On the second page, there is a real time data table that contains temperature and flowrate data from the heat exchanger.

In the upper right corner of the interface is the section labelled Parameter Controls that contains all the input parameters for the heat exchanger. There are two knobs that control the hot and cold flowrates. Alternatively, a user can enter a number for the flowrate in the boxes below the knobs. The flowrates can vary between 0.5 and 5 liters per minute. A user can switch between countercurrent and cocurrent operation by dragging the selector switch to the appropriate side. There also is an input panel for the PID parameters and an input box for the hot water setpoint temperature. If a user wishes to record data, he/she can type a filename in the appropriate box and press the adjacent button to start data recording. The button will turn green once data recording is initiated and the data is stored on the server.

One of the weaknesses of the Java Client Interface was the way it represented the temperature data from the heat exchanger. In the Temperature Data section, both numerical and graphical data are presented so that students can see how the temperatures are changing over time. There is a picture that represents the HT31 double-pipe heat exchanger, and arrows represent the direction of fluid flow when configured for countercurrent operation. Overlaid on this image are six temperature readings from thermocouples placed at six points on the exchanger. Looking at the image, students can see the direction of fluid flow, the location where temperature readings are taken, and can read the numerical temperatures at those points. The graph shows the inlet and outlet temperatures for the hot and cold fluid streams. The graph constantly scrolls while the system is running and shows the last five minutes of temperature data. Students can scale the temperature axis (ordinate) by clicking on the maximum or minimum values and entering a new value. The temperature graph is helpful in determining when the heat exchanger has reached steady state so that students can begin taking measurements.

The Flow Rate Data section contains two scrolling graphs, one for the hot flowrate and one for the cold flowrate. On each graph panel there is also a numerical indicator of the flowrate. The flowrate axis (ordinate) of both graphs is scaleable by clicking on the maximum or minimum values and entering a new value. The flow rate graphs show data for the last three minutes. These

graphs are helpful in detecting when the flowrate has reached steady state. In addition, these graphs also detect irregularities in the control algorithm, which can occur at very low values of the hot water flow rate. This is associated with the operating range limit of the pump, which causes the flow rate to jump between values such as 0.2 and 0.6 L/min when the flow rate is set at 0.4 L/min.

On the second page of the GUI is a data table that contains temperature and flowrate values recorded since the system was turned on. This feature was designed to give students a better estimate of the data that they would be recording to an Excel file. It also enables them to better understand the precision of the measurements they will be making.

USER MANAGEMENT, AUTHENTICATION, & SCHEDULING

On the I-Lab Heat Exchanger Project website (http://heatex.mit.edu), we make use of the Microsoft .NET architecture for web services. The features of .NET make it very simple to collect, manipulate, and display data. The individual pages are ASP.NET web forms, and C# was used as the programming language. In order to provide database functionality for the website, Microsoft SQL Server 2000 was used as the database server. Because of constraints on the equipment available, this database server was forced to reside on the same machine as the web server. Accesses to the SQL server database were performed using ADO.NET.

Users must register to use the system before they will be allowed to perform an experiment. At the time of registration, the user is prompted to choose a Login ID and password, email address, and phone number. Users can also select the team number they belong to at this point. This information is stored in the SQL database. Before a user can perform the experiment, he/she must enter his/her LoginID and Password, which will be checked against existing values in the database.

Students can also sign up for times to perform an experiment on the web. There is a schedule page that shows all the timeslots and who occupies them, if occupied. Students can then select an available timeslot and their name and team will appear on the schedule page.

STUDENT ASSESSMENT

The heat exchanger has been successfully used in a variety of different classes and for different purposes at MIT. It has been used in (1) Transport Processes classes to demonstrate principles of heat transfer, (2) Process Dynamics and Control classes to demonstrate how to tune a piece of equipment in real-time, quantify a response to disturbances, and the general notion of how a real piece of equipment may be abstracted in diagrams and equations, and (3) Chemical Engineering Project Laboratory classes to provide experience in data collection, analysis and presentation, and technical report writing.

Assessments were filled out by the students after each class to evaluate usability, fulfillment of educational objectives, suggested improvements, how beneficial the learning experience was, and how much fun the experiment was to perform. The surveys show favorable responses in general and that the I-Lab experience has improved considerably over time as the interface has become more sophisticated, more tools and options have been added, and software and hardware

problems have been fixed. An example of this improvement can be seen in Figure 6. The assessments show that students have very low tolerance for imperfections in the computer interface or software and hardware problems that may arise during the course of an experiment. Without having the heat exchanger equipment directly in front of them, the students find it harder to fathom that equipment is prone to failure and malfunction.

There is compelling evidence that students' experience with the experiment is a strong function of the context in which the experiment is performed. Students who performed the experiment in the Chemical Project Laboratory class rated their experiences as less beneficial and less fun than those who performed it in other classes. This can be clearly seen in Figure 7. One explanation is that the purpose of the experiment in the Chemical Project Laboratory class is mainly to allow students to practice taking data, analyzing it, and then writing a technical report on their findings rather than explore or test a chemical engineering concept. Therefore, the purpose and context in which the experiment is performed is vitally important to the success of the experiment.

Lastly, the students liked both GUIs (see Table I), but when asked to pick one they preferred the LabVIEW interface (when comparison is made between students who had no problems with either interface).

These findings suggest that web-accessible, remotely controlled laboratory experiments can be valuable in a variety of settings and that the value of the experience, as perceived by the students, depends strongly upon the quality, richness, and reliability of the website and GUI and, in the case of complex experiments, on the ability of the students to gain control of the independent variables at their disposal.

ACKNOWLEDGEMENT

This work was supported in part through the I-Campus Initiative at MIT by a grant from the Microsoft Corp. Conversations with James Trevelyan (Univ. of Western Australia) and Jim Henry (Univ. of Tennessee) were helpful in development of the website.

REFERENCES

[1] Khan, R, A, "Software Architecture for Web-Accessible Heat Exchanger Experiment", SM Thesis, MIT, Cambridge, MA (2002).

[2] Knight, M, "Connecting and Teaching Students via Web Services for an Online Laboratory", MEng Thesis, MIT, Cambridge, MA (2003).

FIGURES

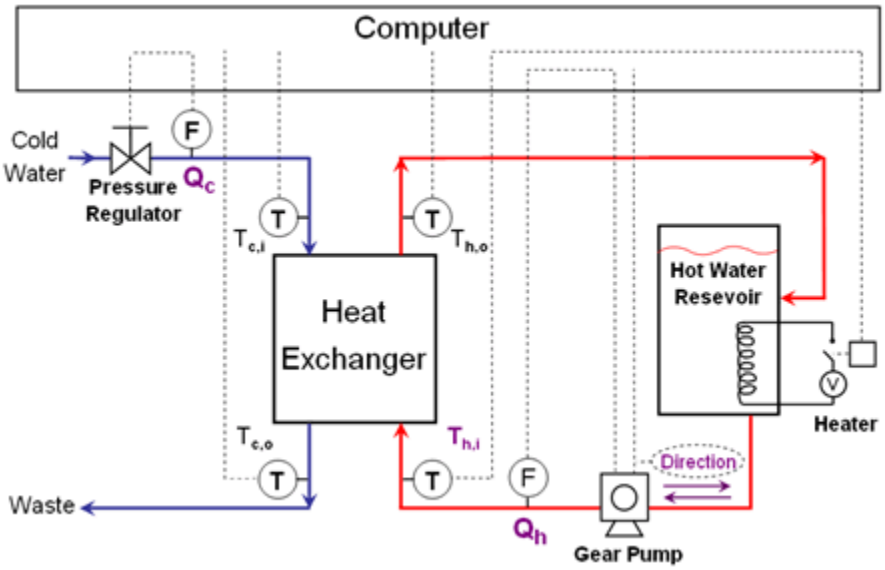

FIGURE. 1
I-LAB HEAT EXCHANGER SCHEMATIC DIAGRAM

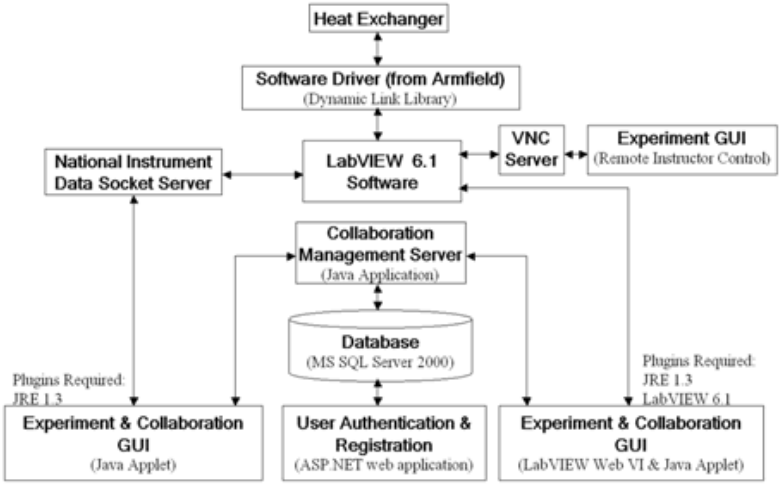

FIGURE. 2

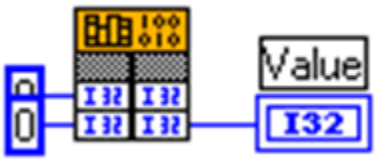

FIGURE. 3
CALL LIBRARY FUNCTION BLOCK DIAGRAM

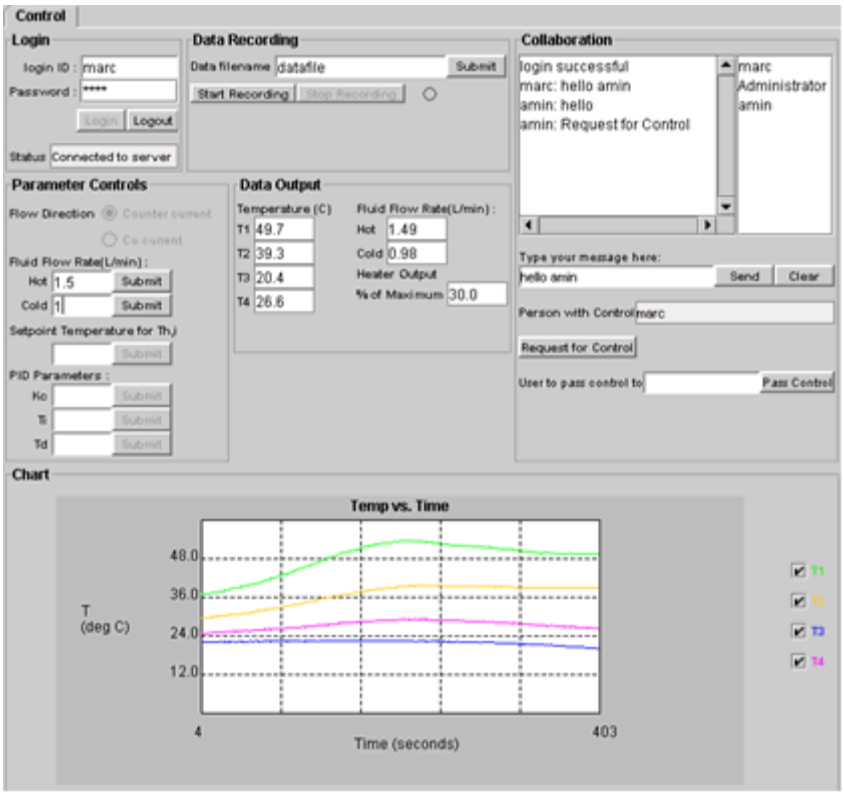

FIGURE. 4
JAVA CLIENT INTERFACE

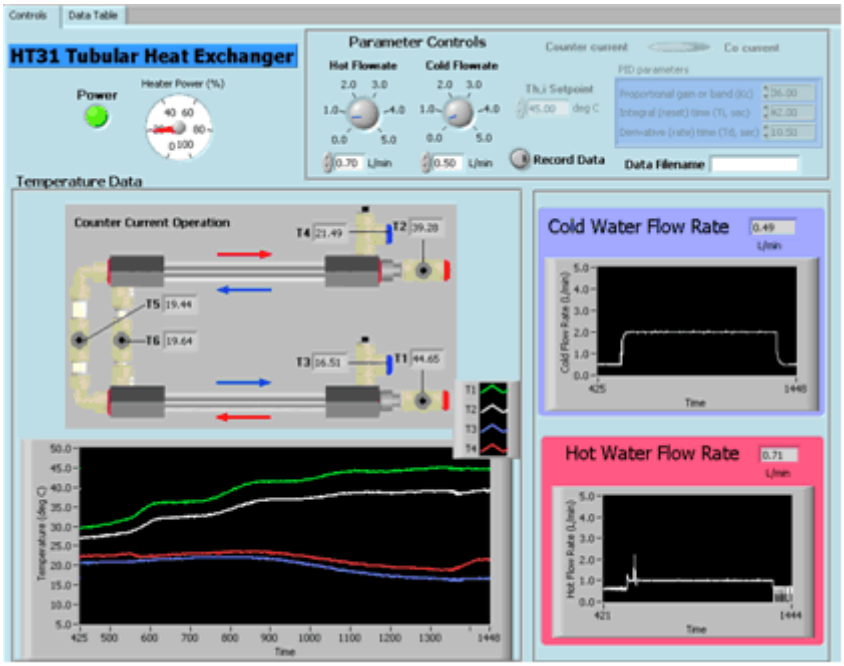

FIGURE. 5
LABVIEW CLIENT INTERFACE

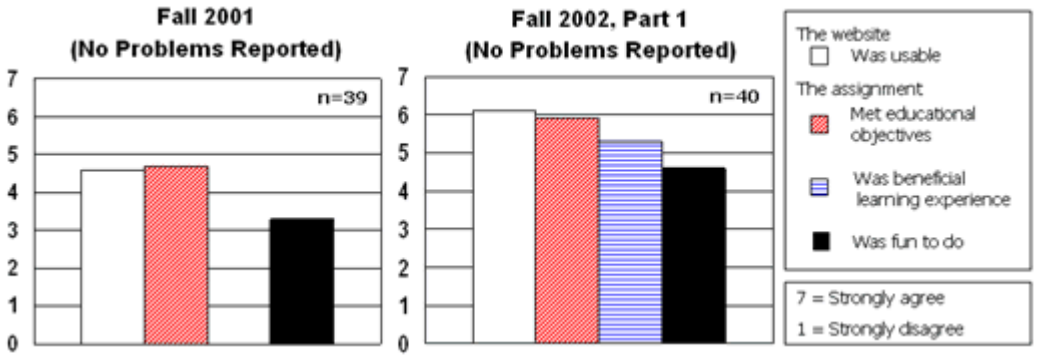

FIGURE. 6
STUDENT ASSESSMENT OF I-LAB HEAT EXCHANGER: COMPARISON OF 10.302 (TRANSFER PROCESSES) IN FALL 2001 AND FALL 2002 USING THE JAVA 2 INTERFACE

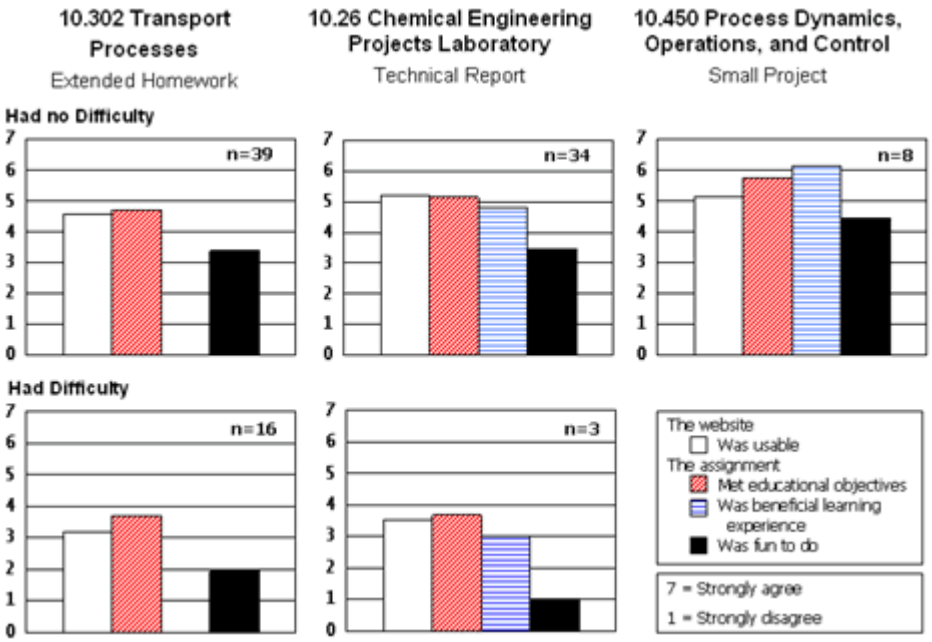

FIGURE. 7
STUDENT ASSESSMENT OF I-LAB HEAT EXCHANGER: COMPARISON OF EVALUATIONS IN DIFFERENT CLASSES

	Reported Problems	
	Yes	No
Usability on the Web:	3.9	5.4
Comparison between GUIs:		
I liked Java 2 (Part 1):	6.0	5.4
I liked LabVIEW (Part 2):	5.1	5.3
If I had to choose one, I prefer:		
Java 2:	5	10
LabVIEW:	2	17
The chat facility for communication was useful:	6.6	5.5
	n = 7	n = 27

TABLE I
STUDENT ASSESSMENT OF I-LAB HEAT EXCHANGER: COMPARISON OF ASSESSMENTS OF DIFFERENT GUIS