

# Connecting and Teaching Students via Web Services for an Online Laboratory

by

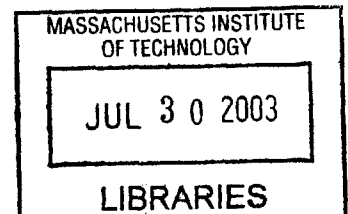
Marc Q. Knight

Submitted to the Department of Electrical Engineering and Computer Science  
in Partial Fulfillment of the Requirements for the Degree of  
Master of Engineering in Electrical Engineering and Computer Science  
at the

Massachusetts Institute of Technology

May 21, 2003

[June 2003]



©2003 Marc Q. Knight. All rights reserved.

The author hereby grants to MIT permission to reproduce  
and to distribute publicly paper and electronic  
copies of this thesis document in whole or in part.

Author: \_\_\_\_\_  
Department of Electrical Engineering and Computer Science  
May 21, 2003

Certified by: \_\_\_\_\_  
Clark K. Colton  
Thesis Supervisor

Accepted by: \_\_\_\_\_  
Arthur C. Smith  
Chairman, Department Committee on Graduate Theses

**BARKER**

# Connecting and Teaching Students via Web Services for an Online Laboratory

by

Marc Q. Knight

Submitted to the  
Department of Electrical Engineering and Computer Science  
on May 21, 2003 in Partial Fulfillment of the  
Requirements for the Degree of Master of Engineering in  
Electrical Engineering and Computer Science

## **Abstract**

The I-Lab heat exchanger is a remotely controlled heat exchanger experiment that has been used on the Internet by students in a variety of chemical engineering courses. The hardware consists of a general service unit, custom designed and fabricated (HT-30XC, Armfield, Ltd., England), on which is mounted a heat exchanger (flat plate, shell-and-tube, or double pipe). Both fluid flowrates, their direction (cocurrent or countercurrent), and the inlet hot temperature are monitored and are under the control of the students. 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 DataSocket server to a Java2 GUI. A database (Microsoft SQL) and ASP.NET web forms are used for registering, authentication, and scheduling. The database is also used for collaboration management software (Java2). 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.

Thesis Supervisor: Clark K. Colton  
Title: Professor of Chemical Engineering

## **ACKNOWLEDGMENTS**

First of all, I would like to thank God for providing me with the strength that has enabled me to make it this far in my life. Second of all, I would like to thank my parents, Bert and Shirley, for providing me with the inspiration and encouragement to continue working towards my masters.

My special thanks to Professor Clark Colton for enabling me to work on this project and for his guidance and supervision over the past year. Also, I would like to thank Dr. Dalzell who was always willing to lend a helping hand.

A lot of this work would not have been possible without Sarah Ibrahim, an exceptional UROP student working on the project. Amin Khan and Richard West, students who were responsible for the initial development of the system, were valuable resources when I encountered problems.

Additionally, I would like to thank John and Steve who were always there when I needed help with something in the lab. Finally, thanks to my family and friends who were supportive of my efforts during the past year.

# TABLE OF CONTENTS

- 1 INTRODUCTION..... 9**
  - 1.1 I-LABS AT MIT ..... 9
  - 1.2 MOTIVATION FOR THE I-LAB HEAT EXCHANGER PROJECT ..... 10
  - 1.3 PROBLEM STATEMENT ..... 11
  - 1.4 THESIS ROADMAP ..... 12
  
- 2 I-LAB HEAT EXCHANGER OVERVIEW..... 13**
  - 2.1 EQUIPMENT DESCRIPTION ..... 13
    - 2.1.1 *HT30XC Service Unit*..... 14
    - 2.1.2 *Heat Exchanger Types* ..... 17
  - 2.2 COMPUTER CONTROL OF THE HEAT EXCHANGER ..... 20
    - 2.2.1 *LabVIEW 6i*..... 21
    - 2.2.2 *DataSocket Server*..... 26
  - 2.3 JAVA CLIENT INTERFACE ..... 28
  
- 3 I-LAB HEAT EXCHANGER IMPROVEMENTS..... 29**
  - 3.1 COLLABORATION UTILITY ..... 32
    - 3.1.1 *Client Application* ..... 33
    - 3.1.2 *Server Application*..... 35
  - 3.2 LABVIEW 6.1 GUI ..... 37
    - 3.2.1 *Parameter Controls*..... 39
    - 3.2.2 *Temperature Data* ..... 39
    - 3.2.3 *Flow Rate Data* ..... 40
    - 3.2.4 *Real-Time Data Table* ..... 40
  - 3.3 PROCESS CONTROL ..... 41
    - 3.3.1 *PID Control Overview* ..... 42
    - 3.3.2 *PID Tuning*..... 43
  - 3.4 USER MANAGEMENT SYSTEM..... 46
    - 3.4.1 *Database Overview* ..... 47

3.4.2	<i>User Registration and Authentication</i> .....	49
3.4.3	<i>Team Management</i> .....	51
3.4.4	<i>Scheduling</i> .....	53
3.4.5	<i>Message Board</i> .....	55
3.4.6	<i>Assessment Questionnaire</i> .....	56
3.4.7	<i>Administrator Features</i> .....	58
3.5	REMOTE WATER RECYCLING .....	62
<b>4</b>	<b>STUDENT FEEDBACK</b> .....	<b>65</b>
4.1	10.302 (TRANSPORT PROCESSES) – FALL 2002 .....	66
4.1.1	<i>Part One – Java Interface</i> .....	66
4.1.2	<i>Part Two – LabVIEW 6.1 Interface</i> .....	69
4.2	10.26 (CHEMICAL ENGINEERING PROJECTS LABORATORY) – SPRING 2003 .....	71
4.3	NOTABLE STUDENT SUGGESTIONS .....	74
<b>5</b>	<b>CONCLUSION AND FUTURE IMPROVEMENTS</b> .....	<b>74</b>
5.1	SUMMARY OF ACCOMPLISHMENTS.....	74
5.2	FUTURE WORK.....	75
	<b>APPENDIX</b> .....	<b>78</b>
	<b>REFERENCES</b> .....	<b>79</b>

# LIST OF FIGURES

FIGURE 2-1: HT30XC SERVICE UNIT (A) SIDE VIEW (B) TOP VIEW .....	15
FIGURE 2-2: (A) DOUBLE-PIPE HEAT EXCHANGER MANUFACTURED BY ARMPFIELD LTD., (B) SCHEMATIC DIAGRAM SHOWING COUNTERCURRENT FLOW .....	18
FIGURE 2-3: (A) SHELL AND TUBE HEAT EXCHANGER MANUFACTURED BY ARMPFIELD LTD., (B) SCHEMATIC DIAGRAM SHOWING COUNTERCURRENT FLOW .....	19
FIGURE 2-4: (A) PLATE HEAT EXCHANGER MANUFACTURED BY ARMPFIELD LTD, (B) SCHEMATIC DIAGRAM SHOWING COUNTERCURRENT FLOW .....	19
FIGURE 2-5: I-LAB HEAT EXCHANGER SCHEMATIC DIAGRAM .....	21
FIGURE 2-6: CALL LIBRARY FUNCTION BLOCK DIAGRAM.....	22
FIGURE 2-7: CALL LIBRARY FUNCTION SETUP DIALOG BOX.....	23
FIGURE 2-8: LABVIEW USER GUI (SERVER CONTROL) .....	24
FIGURE 2-9: DATASOCKET WRITE VI.....	27
FIGURE 2-10: DATASOCKET READ VI.....	27
FIGURE 2-11: JAVA CLIENT INTERFACE (SPRING 2002) .....	29
FIGURE 3-1: I-LAB HEAT EXCHANGER SOFTWARE OVERVIEW: JANUARY 2002.....	30
FIGURE 3-2: I-LAB HEAT EXCHANGER SOFTWARE OVERVIEW: JANUARY 2003.....	32
FIGURE 3-3: JAVA CLIENT INTERFACE (FALL 2002).....	34
FIGURE 3-4: COLLABORATION SERVER.....	36
FIGURE 3-5: LABVIEW 6.1 GUI .....	38
FIGURE 3-6: LABVIEW 6.1 GUI DATA TABLE .....	41
FIGURE 3-7: OSCILLATION IN HOT WATER TEMPERATURE .....	44
FIGURE 3-8: RELATIONSHIP BETWEEN $PB_U$ , $T_U$ AND FLOWRATE.....	45
FIGURE 3-9: LOGIN PAGE.....	50
FIGURE 3-10: USER REGISTRATION PAGE .....	50
FIGURE 3-11: EXPERIMENT HOMEPAGE .....	51
FIGURE 3-12: JOIN A TEAM .....	52
FIGURE 3-13: COURSES, TEAMS, & MEMBERS .....	53
FIGURE 3-14: CALENDAR OF USE.....	54
FIGURE 3-15: A USER'S TIMESLOTS.....	55

FIGURE 3-16: MESSAGE BOARD.....	56
FIGURE 3-17: ASSESSMENT QUESTIONNAIRE .....	57
FIGURE 3-18: QUESTIONNAIRE RESULTS.....	59
FIGURE 3-19: STUDENT COMMENTS.....	59
FIGURE 3-20: MANAGE COURSES AND TEAMS.....	60
FIGURE 3-21: EDIT TIMESLOTS .....	61
FIGURE 3-22: ALL REGISTERED USERS .....	62
FIGURE 3-23: ELECTRONICALLY CONTROLLED VALVES.....	63
FIGURE 3-24: DATA ACQUISITION BOARD & CONNECTOR BLOCK .....	64
FIGURE 3-25: INTERFACE FOR REMOTE WATER RECYCLING .....	65
FIGURE 4-1: STUDENT ASSESSMENT OF I-LAB HEAT EXCHANGER: COMPARISON OF 10.302 (TRANSFER PROCESSES) IN FALL 2001 AND FALL 2002 USING THE JAVA 2 INTERFACE .....	68
FIGURE 4-2: STUDENT ASSESSMENT OF I-LAB HEAT EXCHANGER: COMPARISON OF EVALUATIONS IN DIFFERENT CLASSES.....	73

# LIST OF TABLES

TABLE 3-1: PID CONVERSION FACTORS .....	44
TABLE 4-1: 10.302 PART ONE QUESTIONNAIRE RESULTS .....	67
TABLE 4-2: 10.302 PART TWO QUESTIONNAIRE RESULTS .....	70
TABLE 4-3: 10.26 QUESTIONNAIRE RESULTS .....	72

# 1 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. More and more laboratories are becoming web-accessible as Internet technology is becoming more advanced. The need for these virtual labs fit perfectly into the modern methods of assembly, delivery, and access to educational technology resources adopted by educational institutions around the world.

The I-Campus initiative, a MIT-Microsoft collaborative project, is aimed at using information technology to improve education. One project in particular under the I-Campus umbrella, the Heat Exchanger Project, allows students to remotely control a heat transfer experiment for use in 10.302 and other chemical engineering courses.

## 1.1 I-LABS AT MIT

The I-Lab Heat Exchanger Project (<http://heatex.mit.edu>) is one of several I-Labs under the I-Campus framework at MIT. Other I-Labs include:

- The Microelectronics Weblab (<http://weblab.mit.edu>)
- The Photovoltaic Weather Station (<http://pvbase.mit.edu/index.html>)
- The Flagpole Remote Instrumentation Laboratory (<http://flagpole.mit.edu>)
- The Polymer Crystallization I-Lab (<http://fujicam.mit.edu/>)

These online laboratories can be divided into three groups:

- **Batched Experiment Laboratories:** The Microelectronics Weblab is considered to be a batched experiment laboratory since it is a “load and go” experiment where initial parameters are chosen and data is returned immediately.

- **Sensor Laboratories:** The Photovoltaic Weather Station and the Flagpole Remote Instrumentation Laboratory are considered to be sensor laboratories since they use a publish/subscribe architecture to stream data from their sensors over the web.
- **Hybrid Laboratories:** The Polymer Crystallization I-Lab and the I-Lab Heat Exchanger Project are examples of Hybrid Laboratories. They are a combination of a batched experiment and a sensor laboratory and can be thought of as multiple batched experiment and sensor labs taking place sequentially. For example, a user would first submit data to the server, which would be followed by results being returned from the sensors. Data submissions and updates from the sensors happen multiple times during the course of the experiment. In between data submissions, sensor data is streamed over the web.

These laboratories promote the goals of the I-Campus project at MIT, which are to:

- Monitor real world systems
- Create an interface between engineering and information technology
- Develop an educational environment to provide higher learning for students
- Investigate and implement advances in monitoring technology
- Effectively use virtual environments

## **1.2 MOTIVATION FOR THE I-LAB HEAT EXCHANGER PROJECT**

The main objective of the I-Lab Heat Exchanger Project was to design and implement a real-time, robust and scaleable software system around a laboratory heat exchanger for use in chemical engineering courses to provide students with:

- hands-on experience with a heat transfer experiment for them to compare measured characteristics with theoretical predictions and reflect on discrepancies, limitations, and design criteria.
- remote access to the heat exchanger 24 hours a day, 7 days a week

The I-Lab Heat Exchanger project was designed to be different than a computer simulation and was intended to offer students an enhanced learning experience. On the heat exchanger website, there are numerous pictures of the equipment as well as a QuickTime movie describing its operation. However, when operating the heat exchanger, students sometimes experience unexpected behavior. Furthermore, performing the experiment two separate times with identical parameters will not produce exactly the same results. This unpredictability provides students with a great learning experience, but also makes designing a robust and reliable system more difficult.

### **1.3 PROBLEM STATEMENT**

After the initial implementation of the software system was completed in the spring of 2002, system development has been driven by student suggestions and properties of the equipment. First, the durability of the equipment made it infeasible to have it running 24 hours a day, 7 days a week. The hot water pump only has a lifetime of a couple thousand hours, and the water in the hot water reservoir needs daily recycling to avoid evaporation and fouling. Second, control of the equipment needed to be improved. Students complained when the flowrates and temperatures took a long time to reach steady state, or never reached steady state at all. Third, students complained that the experiment was similar to a simulation and did not know what the equipment looked like, despite the information being available on the webpage.

To reduce wear and tear on the equipment, we decided that students collaborating in teams could perform the experiment. Performing experiments in teams could have the result of enhancing the learning experience for the students. To support this new, team based approach, additional features had to be added to the existing interface and the project website.

These new features would include of a collaboration utility that contained a chat window and a “Pass the Token” capability to determine which student would have control of the equipment. The web services provided at <http://heatex.mit.edu> needed a new registration and signup system that would allow students to be assigned to teams. The website could also include a message board and a live pictures of the equipment.

To address student complaints regarding flowrate and temperature control, we needed to obtain optimal parameters for each of the flowrate and temperature controllers. These parameters were needed for each of the three heat exchanger types (flat plate, shell and tube, or double pipe) in each of the two flow configurations (cocurrent and countercurrent). Additionally, to further differentiate the experiment from a simulation, developing a new interface using LabVIEW 6.1 could be explored.

Furthermore, to make the heat exchanger continuously accessible over a period of several days, there needed to be a method to recycle the water in the hot water reservoir remotely.

These improvements are discussed in the upcoming sections of the thesis.

## **1.4 THESIS ROADMAP**

The purpose of this thesis is two-fold. First, to serve as documentation for the I-Lab Heat Exchanger Project, and second, to explain the details of how these improvements were implemented.

Chapter 2 gives an overview of the I-Lab Heat Exchanger Project as it stood in the spring of 2002, before improvements were made. First, it describes the three heat exchanger types and the service unit on which they are mounted. Second, it describes the USB interface and DLL file used to control the service unit. Furthermore, it describes the use of LabVIEW and Java to develop interfaces for the Heat Exchanger.

Chapter 3 describes all the improvements made to the I-Lab Heat Exchanger during the past year. It describes the implementation of the collaboration capability and the usage of LabVIEW 6.1 to create a new client interface. Furthermore, it gives an overview of the I-Lab Heat Exchanger website and the new web services that were implemented. Finally, it describes how optimal process control parameters were obtained for the various heat exchanger types and how recycling the water in the hot water reservoir can be done remotely.

Chapter 4 summarizes student feedback about the I-Lab Heat Exchanger. Students in MIT courses 10.302 (Transport Processes) and 10.26 (Chemical Engineering Projects Laboratory) performed different experiments on the heat exchanger during Fall 2002 and Spring 2003, respectively.

Chapter 5 concludes the thesis with a discussion on lessons learned and future work

## **2 I-LAB HEAT EXCHANGER OVERVIEW**

### **2.1 EQUIPMENT DESCRIPTION**

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 majority of information contained in this section has been borrowed from the Armfield Manual for the HT30XC service unit [2], and a previous thesis about the I-Lab Heat Exchanger Project [4]. Details on these three heat exchanger types will be discussed in Section 2.1.2.

### ***2.1.1 HT30XC SERVICE UNIT***

The service unit (HT30XC) is a bench-top apparatus on which one of the heat exchangers may be 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.

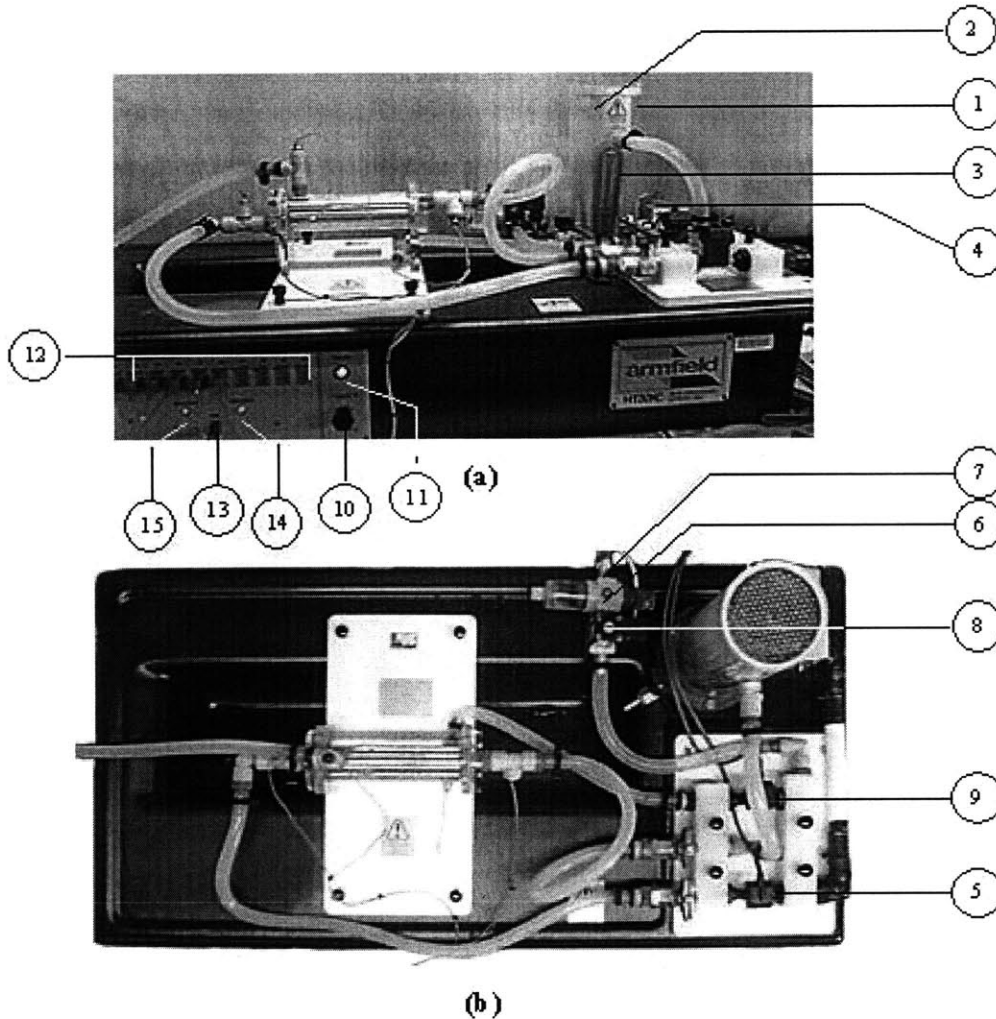
The HT30XC is connected to the computer through a universal serial bus (USB) port and comes equipped with a software driver for the Windows operating system. It is designed to operate from a Windows computer, and all the parameters are computer controlled with no manual intervention other than setting the equipment up and switching it on.

The service unit is equipped with a standard mounting arrangement and service connections for any one of the interchangeable heat exchangers. Figure 2-1 shows several views of the HT30XC Service Unit. The figure shows the HT33 type Shell-and Tube Heat Exchanger with 4 thermocouple sensors mounted on the service unit.

Below are descriptions of the service unit's key components:

**1. Hot Water Reservoir:** This reservoir (1) is situated on top of the service unit. The water in this reservoir is electrically heated. The heating element incorporates an over-temperature thermostat that prevents the water being heated beyond a nominal 85 °C.

Also in the hot water reservoir is a conductivity level sensor (2) that prevents either the heater or pump being turned on unless the reservoir is sufficiently full of water.



**Figure 2-1: HT30XC Service Unit (a) Side View (b) Top View**

**2. Heater:** The electric supply to the heater (3) is modulated by a solid state relay (SSR) which is located inside the plinth base. The modulation signals to the SSR are provided directly by the computer, via the USB interface. This modulation signal is transmitted through a digital channel, which means that the only values that can be passed are 0(heater off) and 1(heater on). This requires the need for a PID algorithm (see Section 3.3) in the controlling software in order to keep the hot fluid inlet temperature steady.

**3. Pump and Flow meter:** Mounted by the side of the hot water reservoir is the gear pump (4), driven by an electric motor, which is used to circulate the water through the heat exchanger and back to the reservoir. The rotational speeds of the motor/gear pump, and hence the hot water flow rate can be controlled from the computer. The hot water flow rate is measured by the in-line flow meter (5) and displayed on the computer. The hot water system includes a strainer on each side of the flow meter, contained within the acrylic mounting blocks. These strainers protect the delicate paddle wheel mechanism of the flow meter from any particles, dirt, hair etc., which may find their way into the water.

Using the computer to control a changeover relay mounted in the plinth base can change the direction of rotation of the pump. This relay reverses the polarity of the electrical voltage applied to the motor, producing a cocurrent or a countercurrent flow.

**4. Cold water pressure regulator:** The cold water flow (the process flow) for the heat exchanger is derived from the local tap water supply. A pressure regulator (6) complete with integral filter/strainer isolates the HT30XC from the minor variations in the pressure of this supply. The cold water supply is connected to the inlet (7) of the pressure regulator. The flow rate through the heat exchanger is then controlled using an electronically driven proportioning solenoid valve (8). Again, this valve is controlled from the computer. A second in-line flow meter (9) measures the cold flow rate.

**5. Flexible tubes:** Flexible tubes are used to connect the circulator to each heat exchanger and quick release fittings allow rapid connection. Red collars identify the hot water connections and a blue collar identifies the cold water connection.

**6. Others:** A panel on the front of the Service unit contains the 'Standby/Enable' switch (10) with 'Control' indicator (11), the 'Emergency Stop' switch with 'Process' indicator and input connectors (12) for up to 10 standard 'k' type thermocouples, labeled T1 to T10. The thermocouples are supplied with the individual heat exchangers and appropriately connected and marked. Also mounted on the panel is the connector (13)

for the USB interface for connection to the computer, and two USB status indicators. A red 'power' LED (14) lights when the unit is connected to the PC and a green 'active' LED (15) lights when the unit has been recognized by the PC. The USB interface is located behind the front panel.

## **2.1.2 HEAT EXCHANGER TYPES**

### **2.1.2.1 BACKGROUND**

Heat exchangers are typically classified according to flow arrangements and type of construction.

There are two types of flow arrangements available

- **parallel-flow or cocurrent:** hot and cold fluid flow in the same directions
- **counterflow or countercurrent:** hot and cold fluid flow in opposite directions

Several types of heat exchangers are available according to type of construction. Some common types are [3]:

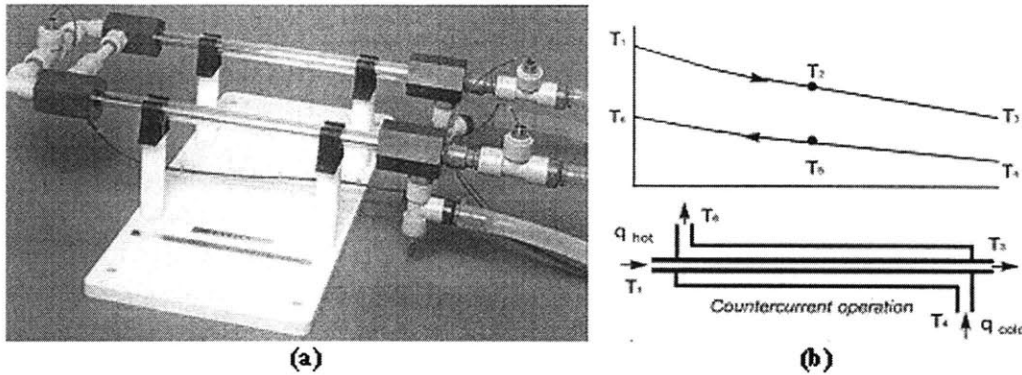
- Concentric tube or double-pipe
- Shell-and-tube
- Flat plate

The HT30XC is able to support all three heat exchanger types and comes with quick disconnect pipe fittings so different exchangers can be connected to the system easily.

### **2.1.2.2 CONCENTRIC TUBE OR DOUBLE-PIPE HEAT EXCHANGER**

This is the simplest type of heat exchanger, which has a long, small-diameter tube placed concentrically within a larger tube (Figure 2-2). One fluid passes through the inner tube,

and the other fluid passes through the outer tube. This type of heat exchanger is capable of handling high pressures and wide temperature differences, but it provides rather poor thermal performance because of a small heat-transfer area.



**Figure 2-2: (a) Double-pipe heat exchanger manufactured by Armfield Ltd., (b) Schematic diagram showing countercurrent flow**

### 2.1.2.3 SHELL AND TUBE HEAT EXCHANGER

Shell and tube heat exchangers (Figure 2-3) consist of a bundle of parallel tubes that provide the heat transfer surface separating two fluid streams. The tube-side fluid passes axially through the inside of the tubes; the shell side fluid passes over the outside of the tubes. Specific forms differ according to the number of shell-and-tube passes, and the simplest form, involves single tube and shell passes. Baffles external and perpendicular to the tubes are usually installed to direct the flow across the tubes, to provide tube support and also to increase the convection coefficient of the shell-side fluid by inducing turbulence and a cross-flow velocity component. The thermal performance of such an exchanger usually surpasses a tubular type but is less than a plate type. The pressure capability of shell-and-tube exchangers is generally higher than a plate type but lower than a tubular type.

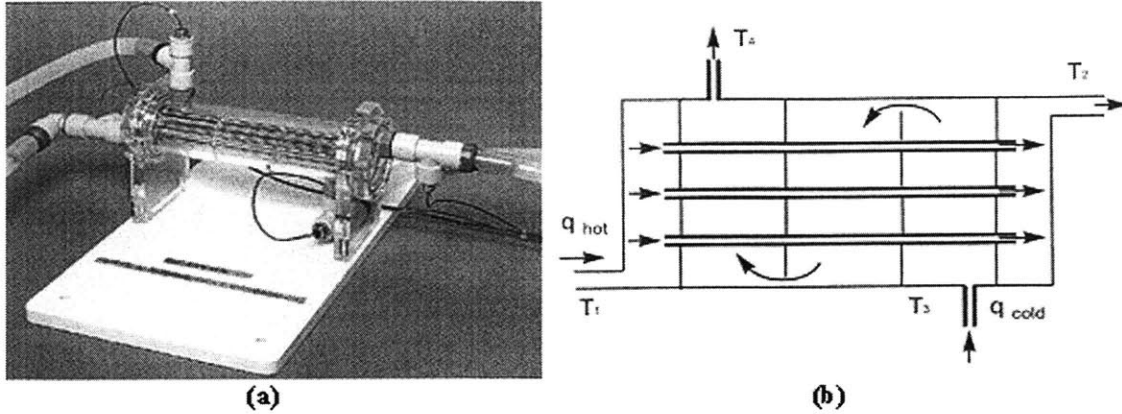


Figure 2-3: (a) Shell and tube heat exchanger manufactured by Armfield Ltd., (b) Schematic diagram showing countercurrent flow

#### 2.1.2.4 PLATE HEAT EXCHANGER

Plate heat exchangers (Figure 2-4) consist of a stack of parallel thin plates that lie between heavy end plates. Each fluid stream passes alternately between adjoining plates in the stack, exchanging heat through the plates. The plates are corrugated for strength and to enhance heat transfer by directing the flow and increasing turbulence. These exchangers have high heat-transfer coefficients and area, the pressure drop is also typically low, and they often provide very high effectiveness. However, they have relatively low pressure capability.

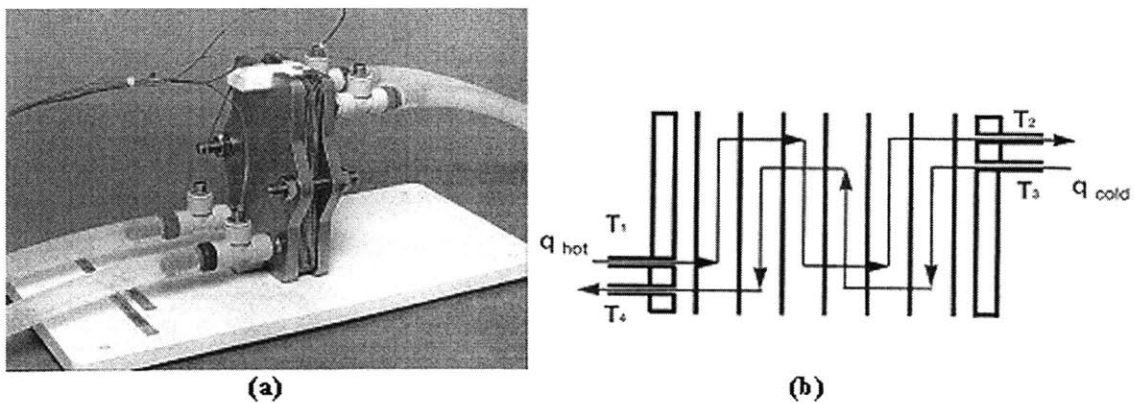


Figure 2-4: (a) Plate heat exchanger manufactured by Armfield Ltd, (b) Schematic diagram showing countercurrent flow

## **2.2 COMPUTER CONTROL OF THE HEAT EXCHANGER**

In order for the heat exchanger to be web-accessible, there needed to be a bridge between the HT30XC service unit and the server. The HT30XC service unit comes equipped with a USB cable and software for controlling the exchanger and for monitoring temperatures from a PC. This software uses a Dynamic Link Library file (DLL) driver, allowing access to one or more particular functions from the controlling program.

The DLL file provides a means to access the input and output parameters of the HT30XC service unit, and is critical for system development since the bundled software package is not designed for web-based control of the unit. If web-based control of the heat exchanger were to be offered, new software would have to be developed that utilized the functions already contained in the DLL file.

Figure 2-5 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.

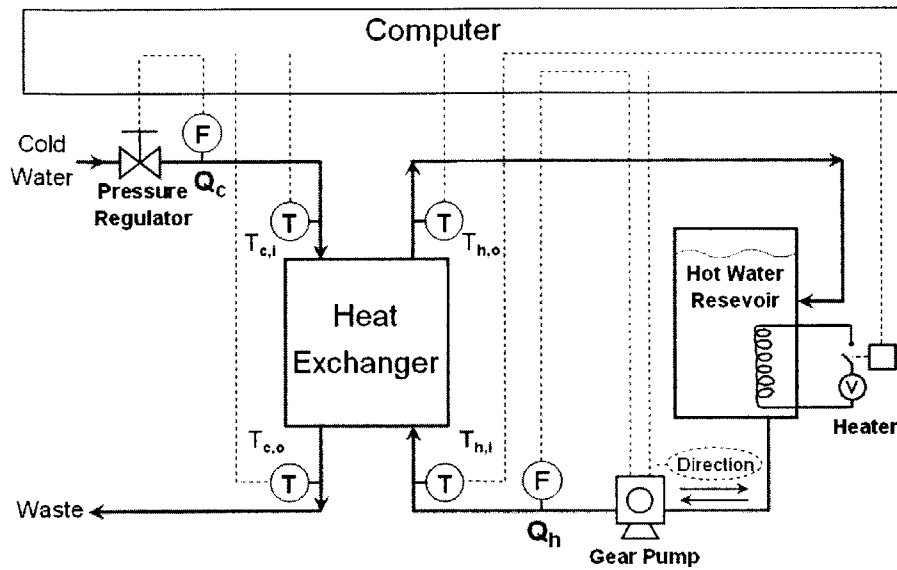


Figure 2-5: I-Lab Heat Exchanger Schematic Diagram

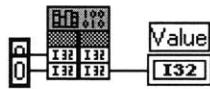
### 2.2.1 LABVIEW 6i

To utilize the functions in the DLL file, the LabVIEW programming language and DataSocket server [7] were used. This enabled data and commands to be transmitted and received via the web.

LabVIEW 6i is a graphical programming language developed by National Instruments that is designed for faster development of programs. The DataSocket server is another product by National Instruments that allows a program to publish and subscribe to data via a specific URI (Universal Resource Indicator), thus making the data web accessible. More detail on the DataSocket server is contained in Section 2.2.2.

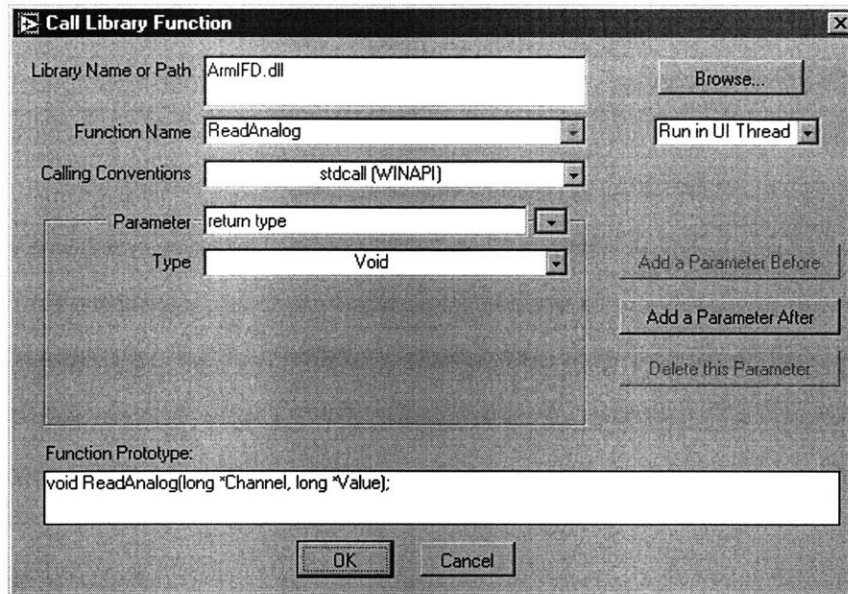
In LabVIEW, block diagrams or VIs (Virtual Instruments) are used to develop a Graphical User Interface (GUI) to monitor and control parameters. This programming language contains a very large library of Graphical Instrument control tools, such as knobs, dials, and charts, which make creating user GUIs 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 2-6. The column on the left corresponds to the function inputs, while the column on the right corresponds to the function outputs. The component labeled Value is an indicator so that the value returned by the function will display on the GUI front panel. 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.



**Figure 2-6: Call Library Function Block Diagram**

Figure 2-7 shows the call library function setup dialog box. Of interest are the Library Name and Function Prototype fields. In the Library Name field, we are able to specify the .dll file that contains the function we wish to use; in this case ArmIFD.dll. The Function Prototype field contains the syntax for the function that we wish to use. In this case, we are trying to read one of the analog data channels coming from the heat exchanger and display the result in an indicator named Value. The heat exchanger has 15 analog channels that can be used to monitor the temperature at any one of the thermocouples or the flowrate at any of the two flowmeters.



**Figure 2-7: Call Library Function Setup Dialog Box**

After using the call library functions to interface with the heat exchanger, the remainder of the user GUI was developed using other LabVIEW control tools. The LabVIEW Instrument Panel is designed to act as a server-side application to control the heat exchanger. Through the use of a sliding control, the user can relinquish control and give control to a client computer via the DataSocket server. The complete LabVIEW interface is shown in Figure 2-8.

The Controls page contains all the controls and indicators to pass values to the equipment and from the equipment through the USB connection. The Power switch on the top turns the equipment on or off. The Power Indicator lights up when the equipment is turned on.

On the left side of the panel is the Input Console. At the top is a switch that switches the control of the equipment between the Administrator (Server) and the user (Client). When in client mode, the Server Administrator doesn't have any control over the parameter values passed to the equipment.

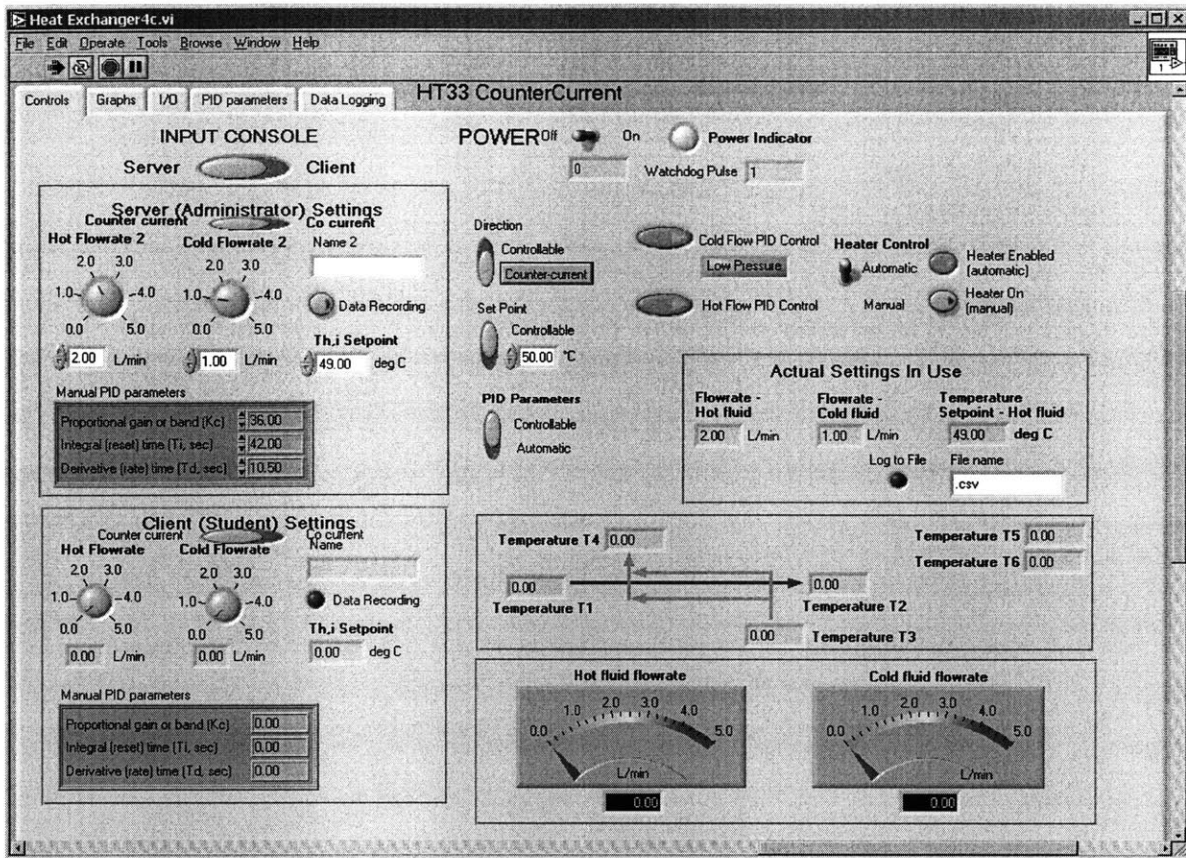


Figure 2-8: LabVIEW User GUI (Server Control)

In client mode, the software listens to particular DataSocket addresses (described in Section 2.2.2) for values of these parameters and passes the values to the equipment as they become available. While in Server mode, the client machines cannot change the parameter values. The values can only be changed from the server. The controllable parameters can be divided into two groups:

**Primary controls** - parameter values that are sent directly to the equipment

- The flow direction (Countercurrent or Cocurrent)
- Hot fluid flowrate: Varying the speed of the pump controls the flowrate of the hot fluid stream. This is done by accessing one of the channels through an Analog Output function call to the .dll.
- Cold fluid flowrate: Controlling a valve using an Analog Output function call regulates the flowrate of the cold fluid stream.

- Heater Control: The heater in the hot water reservoir can only be turned on or off. This heater is used to control the temperature of the hot fluid stream. Although this is a direct control parameter, it is controlled from within the PID (proportional-integral-derivative) algorithm. This is discussed in depth in Section 3.3.

**Secondary controls** - control parameters that are processed within the software and depending on the output, either send a value to the equipment, or perform some other operation.

- Temperature Setpoint: Controlling the heater determines the hot water inlet temperature of the equipment. The heater can only be turned on or off. As a result, a PID algorithm that determines how much of the time the heater is turned on or off was implemented. The algorithm looks backwards in time five seconds and then depending on the PID values, the setpoint value, and the current thermocouple reading at the hot water inlet (received from the equipment), determines the percentage of time during the interval the heater is to be turned on. This keeps the temperature in the hot water reservoir steady.
- PID (Proportional-Integral-Derivative) parameters: These are the parameters used in the PID algorithm to compute how much of the time the heater is turned on or off.
- Data Recording: This consists of a text box to input the filename and a data recording button. The software creates a file with the same name at a location on the server. When the button is pressed, data recording starts and when the button is pushed again, data recording stops.

On the right side of the Controls panel are the parameter values that are passed to the equipment. The administrator can also give limited control capability to the students by disabling some of the controls. This can be done using the selector switches in the middle of the panel. The switches labeled “Direction”, “Setpoint”, and “PID parameters” can be set to either *Controllable* or *Automatic*, depending on the scope of the experiment. If a control is set to *Automatic*, the corresponding control on the client applet becomes gray.

Shown at the bottom right part of the panel are the actual temperature and flowrate values. The temperature values are received from thermocouples mounted at different flow locations. 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)

These are the four standard thermocouple and temperature readings received from the equipment. 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 HT30XC service unit can support a maximum of 10 thermocouples. The flowrate values are received from the two flow meters mounted in the two flow streams with a maximum of 5 L/min. Both the temperature and flowrate readings are accessed through the use of Analog Input function calls.

### ***2.2.2 DATASOCKET SERVER***

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.

### 2.2.2.1 PUBLISHING AND RECEIVING DATA IN LABVIEW

Publishing data to the Internet using the DataSocket server from LabVIEW is made very simple in LabVIEW 6i. LabVIEW provides two functions that interface with the DataSocket server: one to read data, and another to write data.

Figure 2-9 shows the DataSocket Write VI. Data is wired to the *data* parameter in the VI and a string constant specifies the URL parameter. The *URL* in this case corresponds to the DataSocket URI (with a prefix *dstp://*).

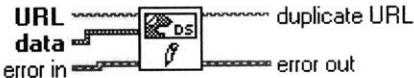


Figure 2-9: DataSocket Write VI

The DataSocket Read VI is shown in Figure 2-10. A string constant specifies the URL parameter. The URL is the address from where the data is read and its type is specified by the type parameter. The VI will wait for an updated value when the parameter *wait for updated value* is set to True. The *ms timeout* parameter specifies how long to wait for a value update.

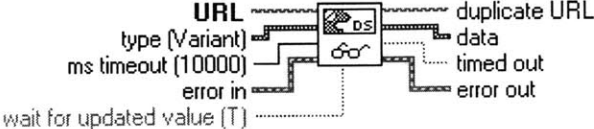


Figure 2-10: DataSocket Read VI

## 2.3 JAVA CLIENT INTERFACE

The Java Client Interface shown in Figure 2-11 has similar parameter controls when compared with the LabVIEW interface described in Section 2.2.1.

In the server connection panel, a user must first press the “Connect” button. This establishes a connection to the DataSocket server and instructs it to begin streaming data to the Java Interface. Once a connection is established, the user will be able to view temperatures and flowrates.

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

- Flow direction (Counter current or Cocurrent)
- Hot fluid flow rate
- Cold fluid flowrate
- Temperature Setpoint
- PID (Proportional-Integral-Derivative) parameters

Users must click the Submit button to update the parameter values. Depending on the experiment, users may not have control over all the parameters. If a user 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 Shell and Tube HT33 Heat Exchanger, the thermocouple locations are T1 through T4; T5 and T6 are not used.

The chart panel contains a scrolling graph that displays the hot inlet temperature (T1). Both the X and Y axes are scaleable.

The data recording panel allows users to save 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. Only after Submit has been pressed, the user can press the Start Recording button. When recording starts, a green light will appear after which he/she can press the Stop Recording button to stop the data recording.

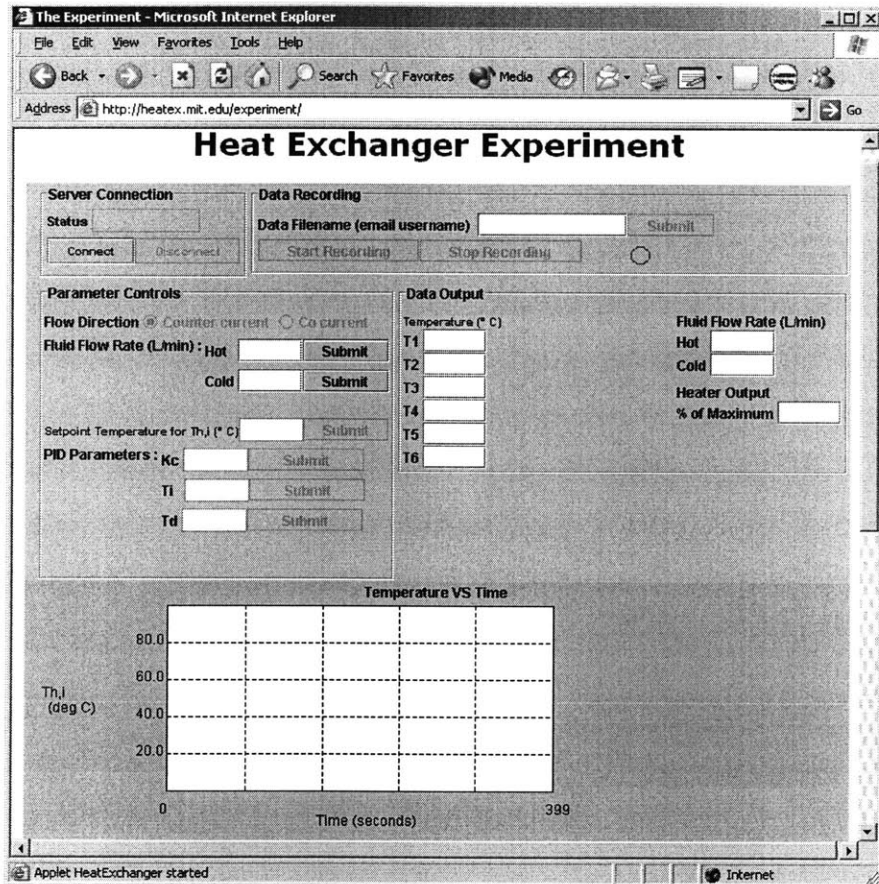


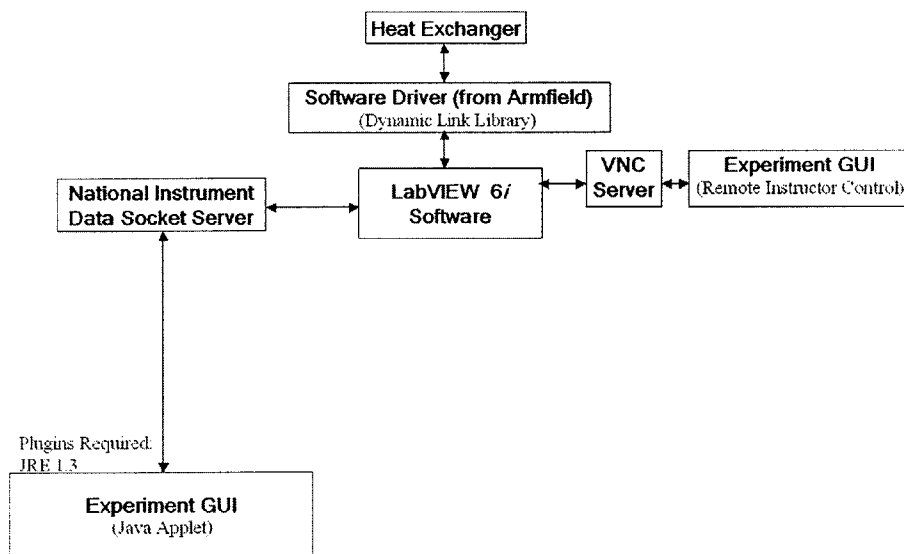
Figure 2-11: Java Client Interface (Spring 2002)

### 3 I-LAB HEAT EXCHANGER IMPROVEMENTS

During the course of work on this thesis, many changes have been made to the existing software architecture of the I-Lab Heat Exchanger. Section 2.2 presented an overview of

the initial software architecture used for the I-Lab Heat Exchanger. This architecture is depicted in Figure 3-1. The initial LabVIEW GUI depicted in Figure 2-8 is at the center of the software architecture since it directly interfaced with the Heat Exchanger via the DLL file provided by Armfield. 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, the Java GUI depicted in Figure 2-11 was used to communicate with the LabVIEW software and the Heat Exchanger via the DataSocket server.

### Software Overview: January 2002



**Figure 3-1: I-Lab Heat Exchanger Software Overview: January 2002**

After the release of LabVIEW 6.1, the software architecture was redesigned and included a new web-accessible LabVIEW GUI and collaboration utility. The redesigned software architecture is shown in Figure 3-2.

Since the inception of the I-Lab Heat Exchanger Project, more students at MIT and from other institutions have been using the equipment over the web. This increased usage has made it important that the website be able to keep track of when experiments would be performed and who would be performing them. Making use of ASP.NET web forms and a SQL Server database, the I-Lab Heat Exchanger website is now able to perform this tracking. Students are now required to register online prior to performing an experiment and can sign up for a vacant timeslot that can be viewed online. If a student signs up for a timeslot, the information is transmitted to the SQL Server database and can be retrieved at a later time.

Also included in the redesigned software architecture is a collaboration utility that allows students to collaborate in teams while performing an experiment. The collaboration utility consists of a chat window via which students can talk, and a “Pass the Token” system through which control over the experiment parameters can be passed between students. The collaboration management server manages this process by coordinating message passing between the client computers. It also handles user authentication via the SQL Server database.

Since LabVIEW 6.1 supported publishing a VI directly on the web, there was no need to have a DataSocket server acting as an intermediary between the web-based application and the underlying LabVIEW software. A new GUI using LabVIEW 6.1 was developed (See Figure 3-5) and was published on the web. To evaluate the usability of the new LabVIEW interface, students in MIT course 10.302 (Transport Processes) performed a two part experiment. For Part 1, the Java GUI was used, and In Part 2, the LabVIEW GUI was used. We found that students preferred using the new LabVIEW interface if they experienced no problems while performing the experiment. Additional results from the 10.302 survey are discussed in Section 4.1.

## Software Overview: January 2003

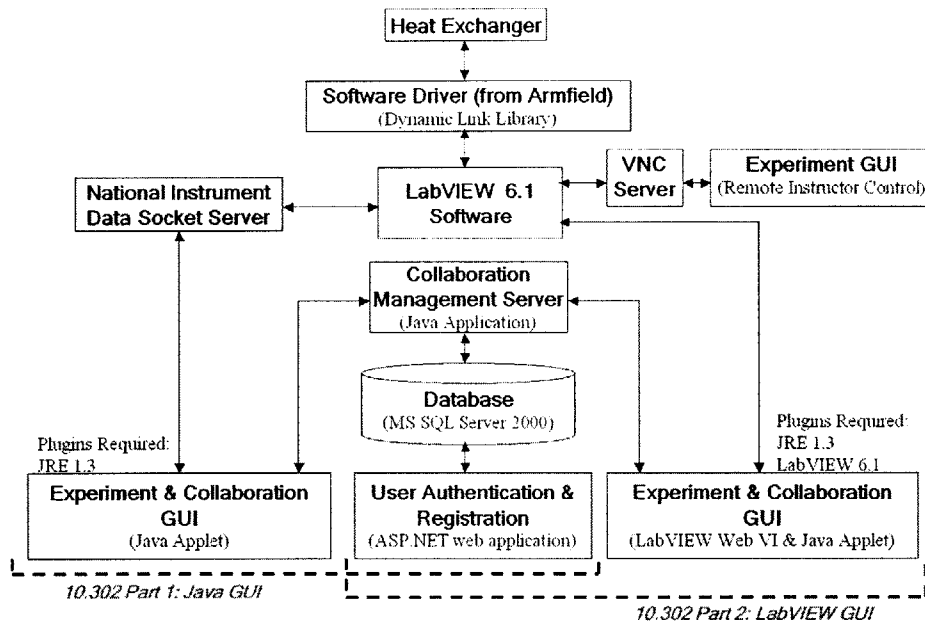


Figure 3-2: I-Lab Heat Exchanger Software Overview: January 2003

### 3.1 COLLABORATION UTILITY

The hot water pump on the Heat Exchanger has a lifetime of a couple thousand hours, making it advantageous to have multiple students performing an experiment at once, thereby reducing wear on the pump. To address this issue, in the fall of 2002, the I-Lab Heat Exchanger was redesigned so that students could perform experiments collaborating in teams.

Students collaborating in teams introduced two problems:

- How would students communicate if they were not working in close proximity to each other?
- How would the system decide who could change the system parameters at any given moment?

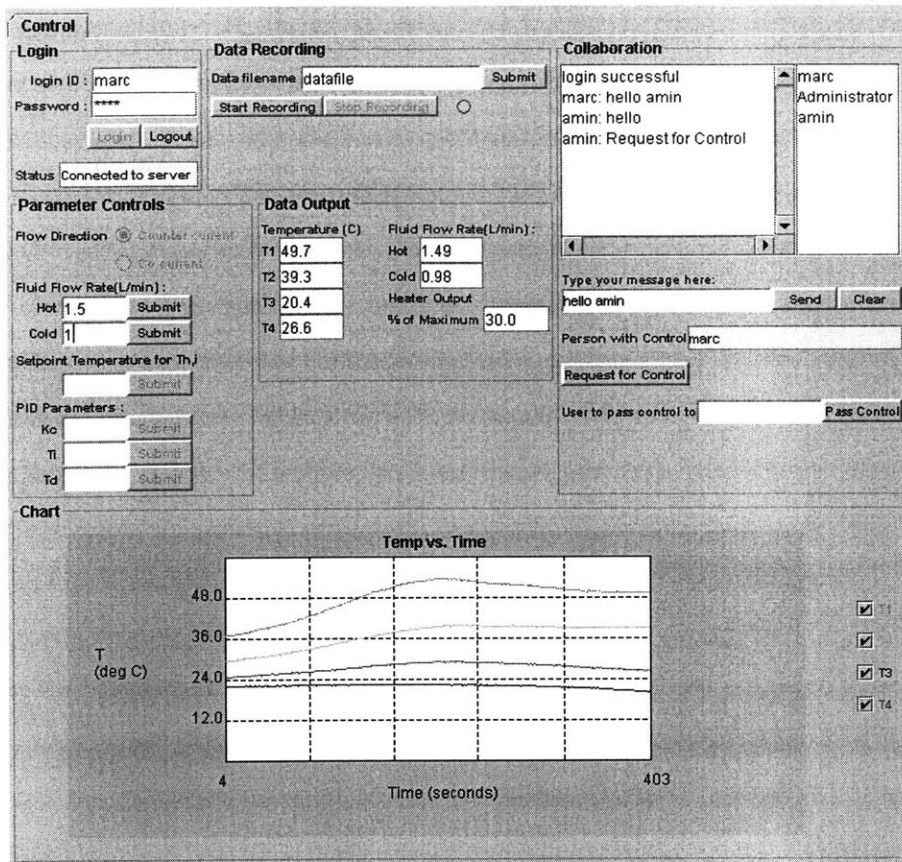
To address these problems, a collaboration utility was integrated into the Java Client Interface that featured a chat window for communication amongst team members and a “Pass the Token” capability so only one team member could control the experiment parameters at any given moment. The chat capability is not as robust as AOL Instant Messenger [1] due to limited time and resources, but does not require a student to download additional software. The collaboration utility is divided into two parts: the server side application that handles user authentication, control, and message passing, and the client application, which is used by students.

### ***3.1.1 CLIENT APPLICATION***

Figure 3-3 depicts the Java Client Application that was developed with the collaboration utility. Besides the collaboration utility, this version of the Java Interface provides user authentication (shown in login panel) and displays more temperatures on the scrolling graph.

To use the Java Interface (client application) a user must first enter his/her Login ID and Password and press the “Login” button. Every user is assigned a Login ID and Password when first registering 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 sends 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.

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.



**Figure 3-3: Java Client Interface (Fall 2002)**

Below this field is the “Pass the Token” component that controls which user can control the experiment 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.

The data output panel and the chart panel exhibit the same behavior regardless whether a user has control over the parameters. The users who cannot control the parameters still can view the data outputs and the charts, but will not be able to record data. Only the user with control will be able to record data.

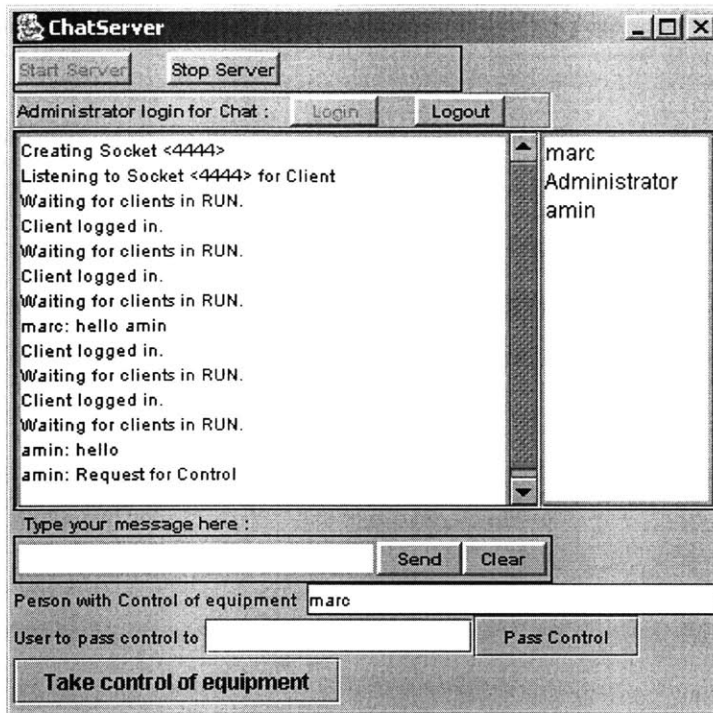
The chart panel contains a scrolling graph that can display the hot/cold inlet and outlet temperatures. Users can select which combination of temperatures they wish to view by selecting the appropriate checkboxes on the side.

### ***3.1.2 SERVER APPLICATION***

The Server Application uses threads to perform multiple tasks. Within one thread, it creates an instance of the *java.net.ServerSocket* class and binds it to a specific server port. It then keeps listening to this port for clients without interrupting other operations. Figure 3-4 shows a screenshot of the collaboration server application. When the “Start Server” button is pressed, the *ServerSocket* is created and the server starts listening to this socket.

If a client connects to the server at that specific port, the server reads from the Socket and tries to authenticate the user. If the client is not authenticated, a denial message is sent to the client containing the reason for denial. If the client is authenticated, a new instance of the *Client* class is created and added to a list of clients. After the new client is created the server keeps on listening to the same port for new clients to log in.

Client authentication is accomplished by connecting to the SQL Server database. More information about the database is contained in Section 3.4.1. JDBC is used to connect to the SQL server database. The JDBC API [5] provides Java programs access to tabular data sources in a large number of formats. While the core API provides a high-level, platform independent interface, it relies on a platform specific implementation to actually access and manipulate the data.



**Figure 3-4: Collaboration Server**

In our case, the Microsoft SQL server 2000 driver for JDBC was used. This driver had to be registered first in order for the JDBC driver manager to know which driver to load. Utilizing the JDBC driver and the appropriate SQL commands, the server checks to see if the Login ID and Password from the client application match the Login ID and Password of any records in the database.

In order to transmit information such as ID's and passwords between the client and server application, the client and server encode this information into messages. For example, a user's Login ID and Password would be contained in a "Login" message that is passed to the server.

The messages that are passed between the server and the client have a specific predefined format. The messages consist of:

- The purpose of the message (the command character)

- Information about the sender (client ID)
- The actual message (body)

The command character is used to distinguish between types of messages. A typical message will have the following format:

Command Character + ClientID + Body of Message

When a message is received at the server, the message is parsed and is divided into several substrings. Then, the message is processed according to the command character that it contains. For example, if the command character is “Login”, then a database access to check the login is performed. Some other key command characters are “Logout”, “Message”, “SeekControl”, “ProvideControl”. When a “Message” is encountered, the message body is rebroadcast to all clients who are currently logged in. This allows all users to see a message that someone has submitted. Likewise, the “SeekControl” command character rebroadcasts a “Request for Control” message. Similarly, the “Provide Control” command character broadcasts a message as to which client should have control over the parameters.

At the bottom of the graphical interface, the Login ID of the user with equipment control is shown. The Administrator has to press the Login button to participate in the chat.

## **3.2 LABVIEW 6.1 GUI**

In November of 2001, National Instruments released the latest version of LabVIEW, Version 6.1. 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.

Creating the Java Client Interface required a substantial amount of time. System development time and system complexity could be reduced if the LabVIEW interface could communicate with the heat exchanger and be made accessible to students over the web. This would eliminate the need for the Java Interface and the use of a DataSocket server. 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 3-5 depicts the LabVIEW GUI that was developed.

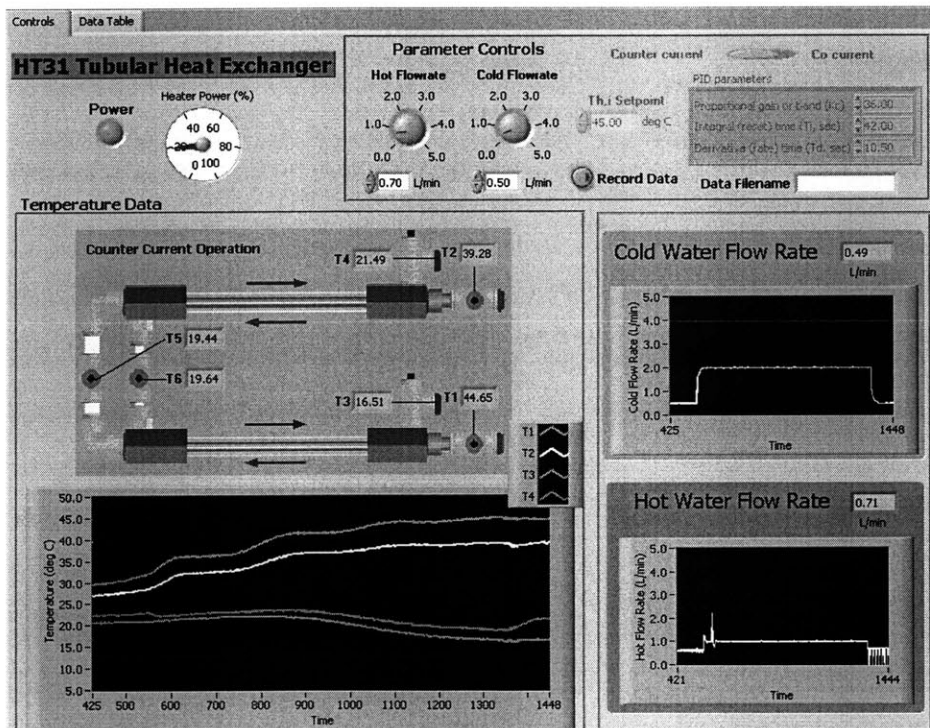


Figure 3-5: LabVIEW 6.1 GUI

The GUI is divided into two pages. On the page shown in Figure 3-5, there are controls and indicators divided into three distinct sections: Parameter Controls, Temperature Data, and Flow Rate Data. On the second page, shown in Figure 3-6, there is a real time data table that contains temperature and flowrate data from the heat exchanger.

### ***3.2.1 PARAMETER CONTROLS***

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.

### ***3.2.2 TEMPERATURE DATA***

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..

### ***3.2.3 FLOW RATE DATA***

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.

### ***3.2.4 REAL-TIME DATA TABLE***

Figure 3-6 shows the data table that is located on the second page of the GUI. The data table 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.

Th = T1	Tm = T5	Tho = T2	Tco = T3	Tcm = T6	Td = T4	Qh (L/min)	Qc (L/min)	Th Setpoint (°C)	Heater Power (%)
23.3	19.5	21.1	15.8	19.8	16.7	1.2	0.5	45.0	0.0
23.4	19.6	21.3	15.8	19.8	16.8	1.2	0.5	45.0	0.0
23.6	19.5	21.6	15.7	19.8	16.8	1.2	0.5	45.0	0.0
23.9	19.5	21.7	15.7	19.8	16.9	1.2	0.5	45.0	0.0
24.1	19.5	21.9	15.6	19.8	17.0	1.2	0.5	45.0	0.0
24.3	19.5	21.9	15.8	19.6	17.1	1.2	0.5	45.0	0.0
24.4	19.5	22.2	15.8	19.8	17.1	1.2	0.5	45.0	0.0
24.9	19.5	22.4	15.7	19.8	17.1	1.2	0.5	45.0	0.0
24.9	19.5	22.4	15.7	19.8	17.2	1.2	0.5	45.0	0.0
24.9	19.5	22.7	15.8	19.9	17.3	1.2	0.5	45.0	0.0
25.0	19.5	22.8	15.7	19.8	17.3	1.2	0.5	45.0	0.0
25.1	19.5	22.9	15.7	19.8	17.3	1.2	0.5	45.0	0.0
25.0	19.5	22.9	15.7	19.8	17.5	1.2	0.5	45.0	0.0
25.0	19.5	23.2	15.7	19.8	17.3	1.2	0.5	45.0	0.0
25.3	19.4	23.3	15.7	19.9	17.5	1.2	0.5	45.0	0.0
25.3	19.6	23.3	15.7	19.8	17.6	1.2	0.5	45.0	0.0
25.2	19.5	23.4	15.7	19.8	17.6	1.2	0.5	45.0	0.0
25.2	19.3	23.4	15.7	19.8	17.6	1.2	0.5	45.0	0.0
25.3	19.4	23.4	15.7	19.8	17.7	1.2	0.5	45.0	0.0
25.3	19.5	23.5	15.7	19.8	17.7	1.2	0.5	45.0	20.0
25.5	19.5	23.6	15.7	20.0	17.8	1.2	0.5	45.0	20.0
25.4	19.5	23.6	15.7	19.8	17.6	1.2	0.5	45.0	20.0
25.6	19.3	23.7	15.7	19.8	17.6	1.2	0.5	45.0	20.0
25.5	19.5	23.7	15.7	19.9	17.8	1.2	0.5	45.0	20.0
25.5	19.5	23.7	15.6	19.8	17.8	1.2	0.5	45.0	40.0
25.6	19.5	23.8	15.7	19.8	17.7	1.2	0.5	45.0	40.0
25.6	19.6	23.9	15.8	19.9	18.0	1.2	0.5	45.0	40.0
25.5	19.6	23.8	15.6	19.9	17.8	1.2	0.5	45.0	40.0
25.4	19.5	24.0	15.7	19.8	18.0	1.2	0.5	45.0	40.0
25.5	19.4	24.0	15.6	19.8	17.9	1.2	0.5	45.0	50.0

Figure 3-6: LabVIEW 6.1 GUI Data Table

### 3.3 PROCESS CONTROL

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.

### ***3.3.1 PID CONTROL OVERVIEW***

A PID controller is a type of feedback controller that generates an output that causes some corrective effort to be applied to a process so as to drive a measurable process variable towards a desired value known as the setpoint. In our case, one process variable is the actual hot water inlet temperature and the setpoint is the desired value for the hot water temperature. The controller determines its output by observing the error between the setpoint and a measurement of the process variable. Its mission is to eliminate the error automatically.

PID stands for Proportional, Integral, Derivative. These designate the three modes that determine the controller output. These equations show the relationships:

- P      Proportional Band =  $100/\text{gain}$
- I      Integral =  $1/\text{reset}$                       (seconds)
- D      Derivative                                      (seconds)

With proportional band, the controller output changes in proportion to a change in the measured error. The greater the error, the greater the control effort; and as long as the error remains, the controller will continue to try to generate a corrective effort. However, with only proportional control, the controller will generally leave a steady state error (offset) between the setpoint and the process variable after it has finished responding to a setpoint change or a load.

The introduction of integral control resolves this problem. An integral controller generates a corrective effort proportional not to the present error, but to the sum of all previous errors. The controller itself is the same, but the addition of the integral action in the process makes the controller more effective.

However, with proportional band and integral control, the controller still takes too long to compensate for a load or a setpoint change. In order to improve performance, a

derivative controller can be added that generates a control action proportional to the time derivative of the error signal. The basic idea behind derivative control is to generate one large corrective effort immediately after a load change in order to begin eliminating the error as quickly as possible.

Equation 3-1 shows how the proportional, integral, and derivative modes are combined to produce the controller output ( $u$ ).  $K_P$  is the gain value,  $K_I$  is the reset time (seconds),  $K_D$  is the derivative time (seconds), and  $e$  is the error value. The error is calculated by subtracting the process variable from the setpoint variable

$$u = K_P e + K_I \int e dt + K_D \frac{de}{dt}$$

**Equation 3-1: PID Controller Output**

### **3.3.2 PID TUNING**

In order to find the optimal PID parameters for the gain, reset time, and derivative time, the Zeigler-Nichols Closed Loop tuning method was used. There were three separate controllers for which optimal PID parameters needed to be found. They consisted of the hot water temperature controller, the cold water flowrate controller, and the hot water flowrate controller. In order to perform the tuning, the following steps (taken from the National Instruments PID Manual [6]) were followed:

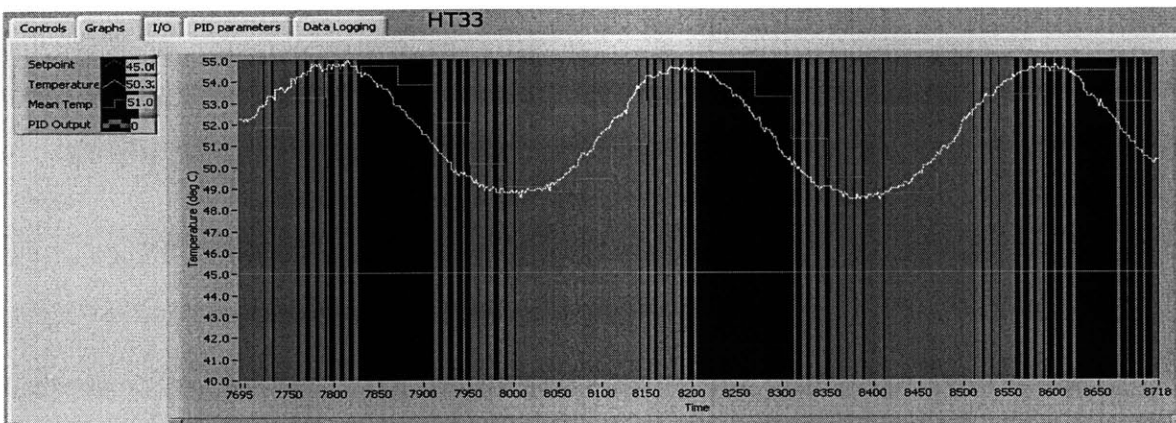
1. Set both the rate(derivative time) and reset on your PID controller to 0.
2. Carefully increase the proportional gain ( $K_P$ ) in small steps. Disturb the loop after each step by making a small change in the setpoint. This will cause the process variable to oscillate. Keep making changes until the oscillation is perfectly sustained, neither growing nor decaying over time.
3. Record the controller proportional band  $PB_u$  as a percent, where  $PB_u = 100/K_P$
4. Record the period of oscillations as  $T_u$  in minutes.

- Multiply the measured values by the factors shown in Table 3-1, and enter the new tuning parameters into your controller. The table provides the proper values for a quarter-decay ratio.

PB (percent)	Reset (minutes)	Rate(minutes)
$1.67PB_u$	$0.50T_u$	$0.125T_u$

**Table 3-1: PID Conversion Factors**

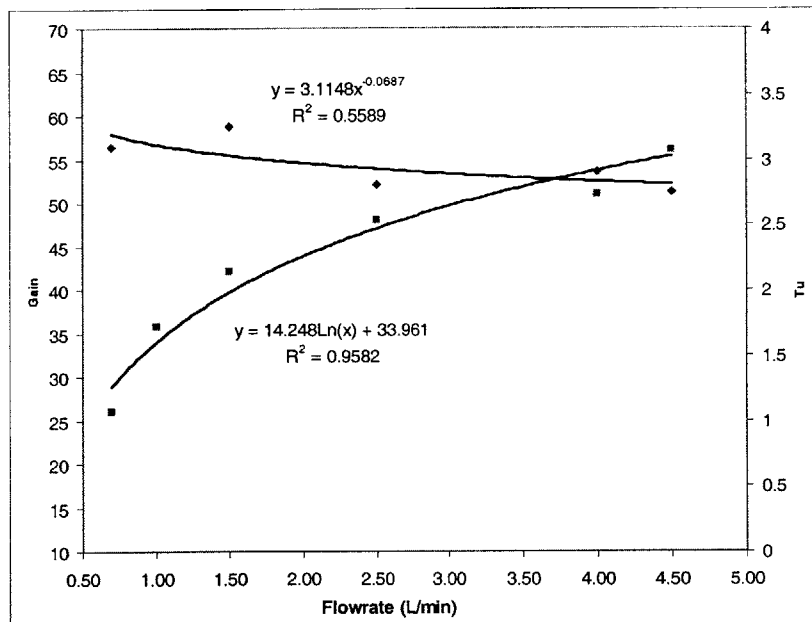
Figure 3-7 illustrates a sustained oscillation in temperature (process variable) after a disturbance in the hot water temperature setpoint. .



**Figure 3-7: Oscillation in Hot Water Temperature**

The optimal parameters 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. To address this problem, optimal values of  $PB_u$  and  $T_u$  were calculated over the range of the hot water flowrate. Optimal values were calculated over the useful range, in 0.5 L/min increments. Curves were fitted to a plot of  $PB_u$  and  $T_u$  versus flowrate to determine a relationship between the flowrate and these parameters. For the double pipe heat exchanger, the relationship between flowrate and  $PB_u$  was logarithmic. The relationship between flowrate and  $T_u$  was exponential. These relationships can be seen in Figure 3-8. The functions for  $PB_u$  and  $T_u$  were multiplied by the conversion

factors in Table 3-1, to obtain the optimal PID settings. This method actually calculates the optimal PID functions that provide the optimal PID parameters at different values of the flowrate.



**Figure 3-8: Relationship between  $PB_u$ ,  $T_u$  and 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). Without these PID settings, the heat exchanger is hard to control and produces unreliable results. If future changes are to be made to the heat exchanger, new PID parameters must be found.

### 3.4 USER MANAGEMENT SYSTEM

Until the spring of 2002, only chemical engineering students at MIT had used the I-Lab Heat Exchanger. The successful offering of the I-Lab Heat Exchanger to students enrolled in the ChE 354 (Transport Processes) course at the University of Texas at Austin proved that the heat exchanger could be offered in courses outside of MIT. Following in the footsteps of the microelectronics weblab, the I-Lab Heat Exchanger has the potential to be a tool that can be used at many universities with students collaborating on different experiments.

The collaboration system allows students to work in teams, thus reducing the amount of time needed for a class to complete an assignment and potentially improving the overall learning experience. However, the team oriented structure introduces more complexity to the services that I-Lab Heat Exchanger website needed to provide. As a result, the web services needed to be able to identify and authenticate teams of students as well as individuals.

It was desirable to make management of the system as simple as possible so that professors could make necessary modifications the website instead of delegating the task to an RA or undergraduate student who would use numerous programs on the server to complete the same task. An interface was developed that allowed a system administrator to enter information about the courses that will be using the heat exchanger, and how many teams of students each of these courses will have. The administrator can select the timeslots for which the equipment will be available, and can keep track of all the users who have registered on the system. There can be multiple website administrators, allowing a professor, RA, and an undergraduate student to all have access to these features.

### ***3.4.1 DATABASE OVERVIEW***

In order to provide database functionality, Microsoft SQL Server 2000 was used as the database server. Due to constraints on the equipment available, this database server was forced to reside on the same machine as the web server. Due to the limited number of simultaneous accesses that our server receives, this has proven not to be a problem.

The user management system consists of eight tables (Courses, CourseTeams, Schedule, Students, Teams, Times, Messages, and Questionnaire) that contain all the information needed by the various web forms that comprise this part of the application. Accesses to the SQL Server database are performed using ADO.NET and the SQL Server Enterprise Manager is used to manage data in the seven tables. The Enterprise Manager allows the system administrator, when present at the server computer, to view and modify data values in any one of the tables. This is relatively easy to do if the system administrator is very familiar with the workings of the Enterprise Manager. However, in many instances, an administrator may wish to modify the database without having any knowledge of its inner workings. The Administrator Features discussed in Section 3.4.7 provide this functionality.

All of the database tables have a HashCode field that uniquely identifies each entry into the data table. Some of the tables use this value as its primary key, which prevents two data entries in a table from having the same HashCode. This eliminates the problem of having duplicate data in tables and also makes for easier programming.

Below are descriptions of the various tables contained in the SQL database.

#### **Courses**

- The Courses table stores the name of a course in the CourseName field, and also keeps track of the number of courses in the NoTeams field. The CourseName field is of type varchar and the NoTeams field is of type int.

#### **CourseTeams**

- The CourseTeams table contains fields CourseName, TeamNumber, and Num. The TeamNumber field contains a character based team identifier or name, and the Num field contains the team number. The Num field is of type int, and the remaining fields are of type varchar.

### **Schedule**

- The Schedule table stores information about a user when he/she signs up for a particular timeslot. The HashCode field stores a hash of two other fields, Date and Time, which store the Date and Time submitted by the user, respectively. Also included are the UserID of the user who produced the entry and the Team that they are associated with, if any. All fields are of type varchar.

### **Students**

- The Students table contains information about every user who has registered to use the system. Every user is assigned a unique UserID, which identifies them in the Schedule, Teams, and Messages tables. The IsAdministrator field indicates if this user will have administrator privileges on the system. All fields are of type varchar.

### **Teams**

- The Teams table stores information about the teams that each user is associated with. The HashCode field contains a hash of the UserID and the CourseName field, to prevent a user from belonging to two teams in the same course. The CourseName and TeamNumber fields are the same as those in the CourseTeams table, and the Team field is the same as that in the Schedule table. All fields are of type varchar.

### **Times**

- The Times table contains a listing of the times to be displayed on the Schedule page. Its fields are of type varchar.

### **Messages**

- The Messages table stores information about a post that is made to the Message Board. Information such as the name of the poster, the Subject, Time, and the Message are stored in the corresponding fields. To eliminate the problem of having duplicate posts, the HashCode field contains a hash of

the Message text and UserID. All fields in the Messages table are type varchar.

### **Questionnaire**

- The Questionnaire table stores data from student responses to the online assessment questionnaire. The data for each question is stored in a corresponding field of type int. The table also contains a HashCode field so the same student cannot submit a questionnaire multiple times.

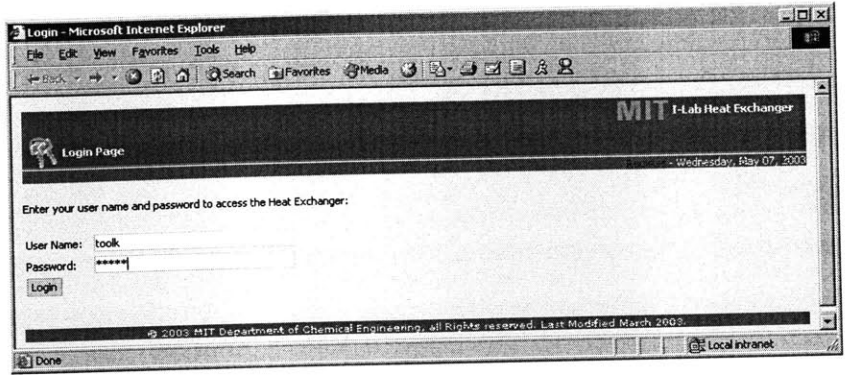
In order to perform operations on data in the tables, tables are first loaded into a DataSet object. A DataSet can be thought of as an in-memory relational database, as it contains a collection of DataTable objects. Another object, which is data provider specific, handles the process of filling the DataSet with data from a database, and posting changes back to that database. This object is a data adapter, and each data provider contains its own version. In this instance, the SqlDataAdapter is used. The data adapter uses command objects (SelectCommand, InsertCommand, etc..) to retrieve data from a database, and later to post changes (inserts, updates, or deletes) back to it. The use of the DataSet and SqlDataAdapter reduces traffic to the SQL database since data is manipulated locally on the client computers.

### ***3.4.2 USER REGISTRATION AND AUTHENTICATION***

Like all the pages in the user management system, the user registration and authentication pages were developed using ASP.NET. Microsoft Visual Studio .NET was used as the form editor and all code was programmed in C#. Both pages are HTML web forms pages that have an .aspx extension. The C# code files have an extension of .aspx.cs.

Figure 3-9 shows the initial login page that authenticates all users. The user is prompted for his/her username and password or has the option to register if this is their first time using the system. If a user is able to successfully authenticate with his/her username and password, the server sends a cookie to the user's computer and sets this cookie value to

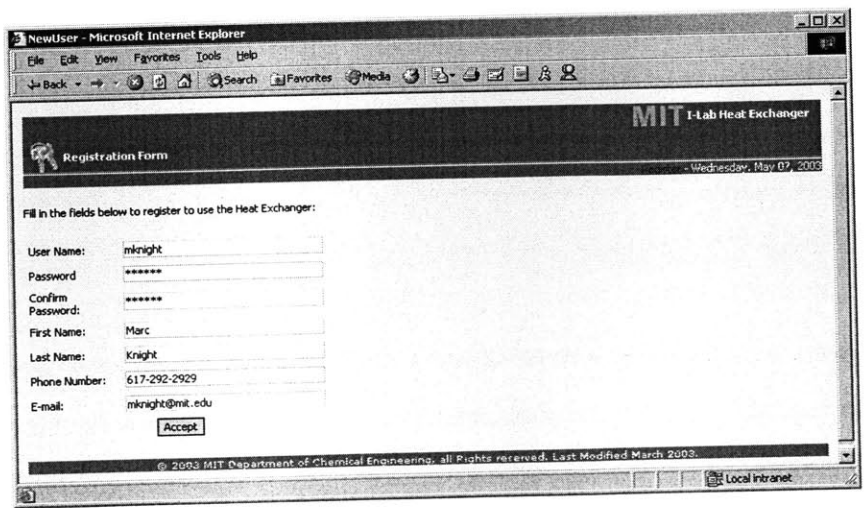
his/her UserID that is stored in the database. Also included in this cookie is information about administrator privileges. The system is able to know whether a user has access to a particular page by tracking this cookie.



**Figure 3-9: Login Page**

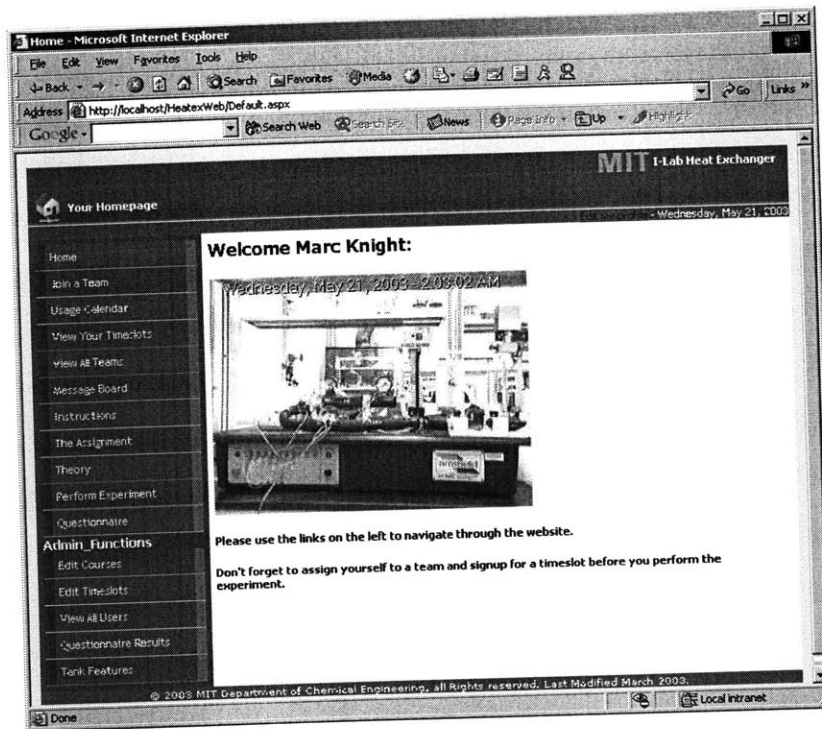
Figure 3-10 shows the user registration page. All fields are required on this page and are entered into the Students table after the Accept button is pressed. To ensure a user enters data in all fields and that this data is in the correct format, three types of validation controls were used. They include:

- A RequiredFieldValidator to ensure that no data field is left empty.
- A CompareValidator to ensure that the two password fields match.
- A RegularExpressionValidator to ensure that the user enters a valid phone number or email address.



**Figure 3-10: User Registration Page**

Once the data is validated, a connection to the SQL Database is opened, the data is sent, and the connection is closed. If there is an error, the user will be notified. An error usually occurs if the same user tries to register on the system twice. After authentication, the user is directed to the homepage.



**Figure 3-11: Experiment Homepage**

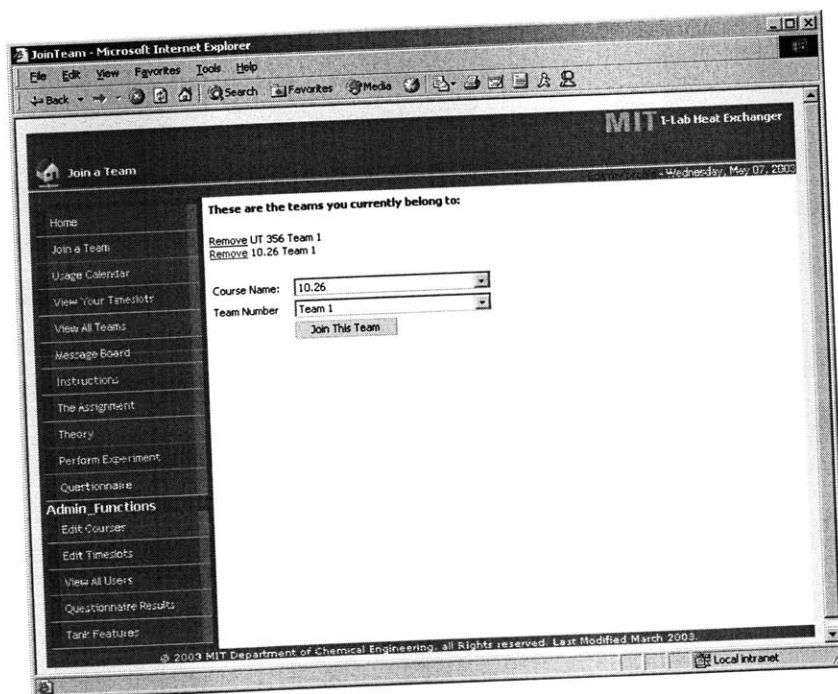
The experiment homepage is depicted in Figure 3-11 and contains a live picture of the equipment. Snapshots are taken from a video camera every 30 seconds and are delivered to this page. Although students may not notice any differences in the images while they are performing an experiment, it allows them to familiarize themselves with the equipment.

### **3.4.3 TEAM MANAGEMENT**

Movement toward a team-based usage of the I-Lab Heat Exchanger has put a priority on the system being able to manage users and teams efficiently and effectively. For

example, users need the flexibility to be able to join multiple teams and see which students belong to other teams.

Figure 3-12 shows the page on which a user can join particular teams. Users select the appropriate course name and team number from the dropdown menus and press the button labeled “Join this Team”. If the user is able to successfully join the team, the team appears under the label “These are the teams you currently belong to”, as shown in Figure 3-12. If a user already belongs to another team in the same course, they are prompted with an error message. When a user joins a team, a new entry in the Teams table is created, thus each line in the Teams table is an association between a particular user and team. If the same user were to join another team, a new entry would be created in the table. Clicking the Remove link to the left of an already joined team causes the proper SQL statement to execute, thus removing the corresponding entry from the database.



**Figure 3-12: Join a Team**

Figure 3-13 shows the page on which users can view members of any team for any course that is using the I-Lab Heat Exchanger. By changing the value in the dropdown list, users

can select the course that they are interested in. A `DataList` control communicates with the `CourseTeams` table and generates an entry for each team that is present for the selected course. Inside each `DataList` entry is a nested `DataGrid`, which retrieves information about the corresponding team members from the `Teams` and `Students` tables. The result is seen in the figure. Email can be sent to any user by clicking on the “Send Mail” link next to his/her name.

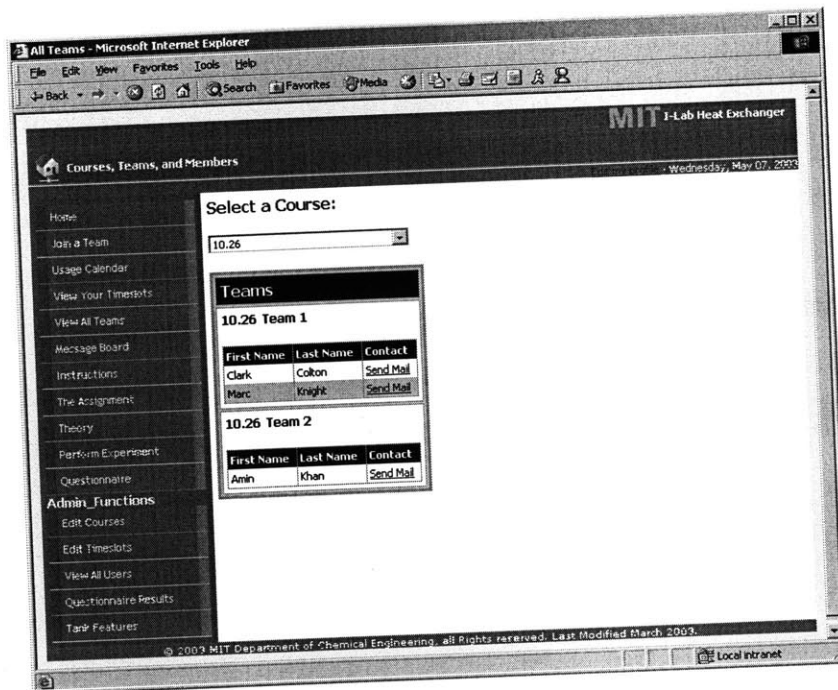


Figure 3-13: Courses, Teams, & Members

### 3.4.4 SCHEDULING

Figure 3-14 shows the page through which users can sign up for timeslots on various days and times. The page uses a calendar control, which facilitates easy selection of dates in the current year. The corresponding `DataGrid` next to the calendar lists the timeslots on the selected date for which the heat exchanger may be available. If a particular timeslot is not available, “Not Available” will appear in the Occupied By field. Otherwise, the Occupied By field is empty, contains a team name, or contains the name

of an individual. To signup for a timeslot, a user needs to press the Select link for the desired timeslot. After clicking the link, the user is prompted whether he/she wants to sign up for the slot. If the user clicks the “No” button, the request is cancelled. If the user clicks the “Yes” button, the request is sent to the server and an entry is added to the Schedule table. If there is an error, the user is prompted with an error message. When signing up for a timeslot, a user must select the correct team number from the dropdown list. After the signup process is complete, the team number will appear in the Occupied By field. If a user does not want to signup their team, they can select None from the dropdown list and their name will appear in the Occupied By field.

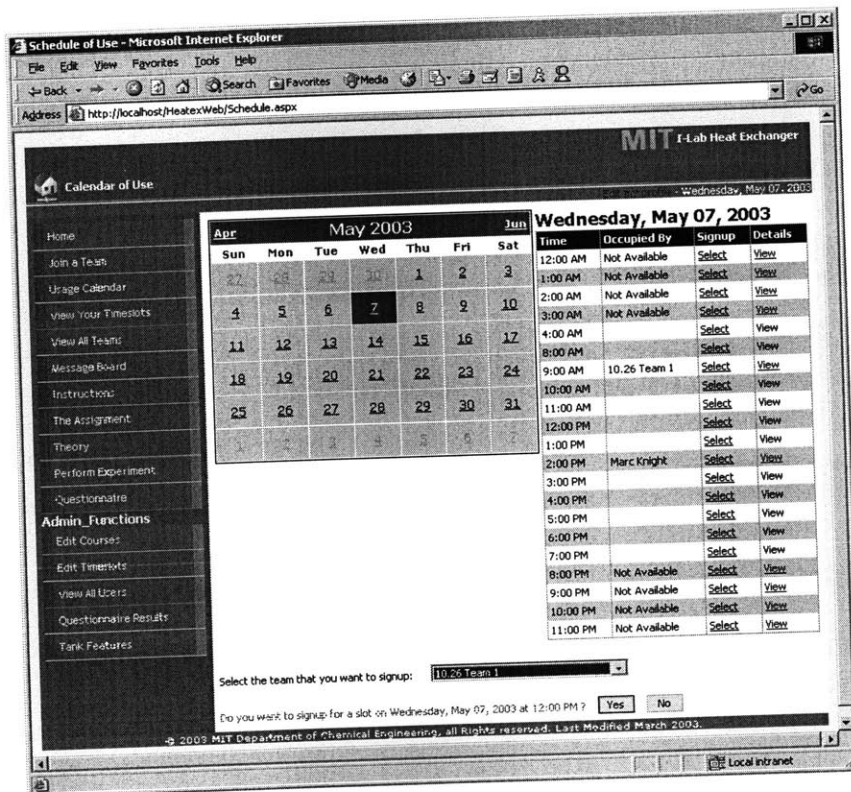


Figure 3-14: Calendar of Use

If a user wants to find out what timeslots he/she has signed up for, they can visit the page shown in Figure 3-15. This page contains a listing of timeslots that a user or other members his/her teams have signed up for. If a user wishes to remove a particular timeslot, he/she can click on the corresponding remove link.

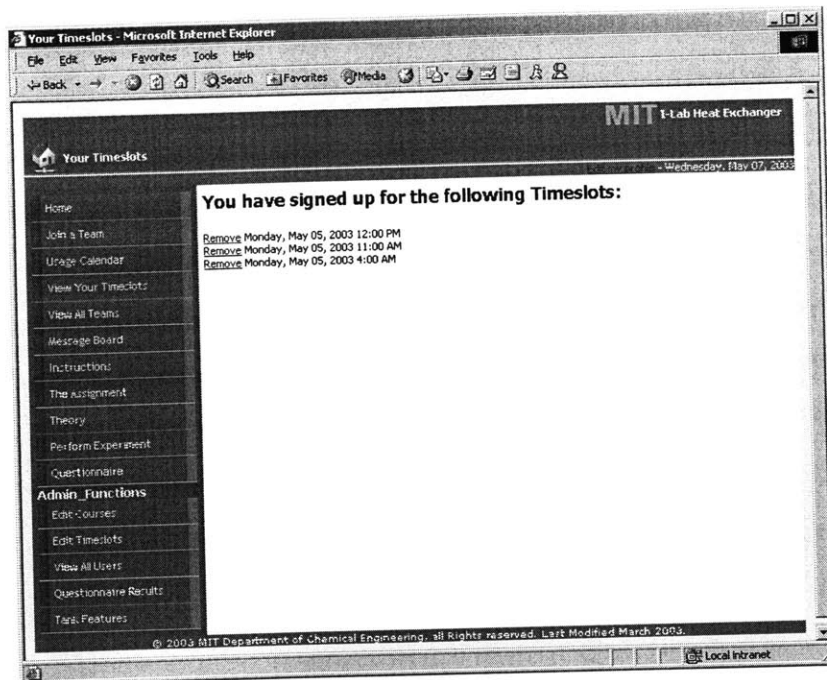
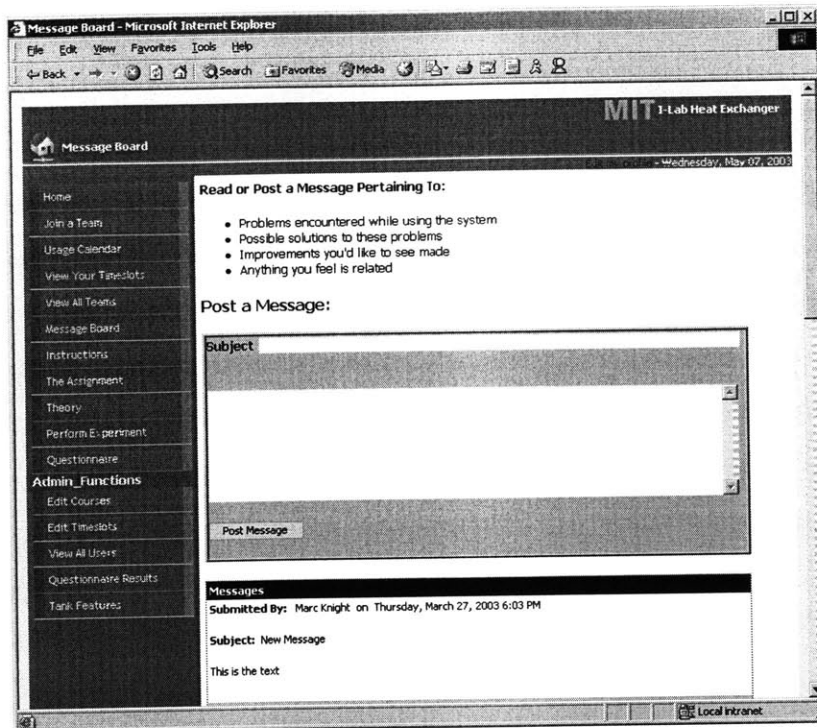


Figure 3-15: A User's Timeslots

### 3.4.5 MESSAGE BOARD

The collaboration utility provides users who are currently accessing the equipment with a method of communication. Often students talk about problems with the experiment and questions about the assignment. Those students may wish to communicate with other teams performing the same experiment. The message board page shown in Figure 3-16 allows users to communicate with other who may not be performing the experiment at the same time. By entering a subject and text for the message and clicking the button labeled "Post Message", the user is able to add his/her message to the DataList that contains previous messages submitted by other users. If a student encounters a problem while running the experiment that is not addressed in the instructions, it is possible that he/she may find a solution in the message board.



**Figure 3-16: Message Board**

### **3.4.6 ASSESSMENT QUESTIONNAIRE**

After every student performs an experiment on the I-Lab Heat Exchanger, they are asked to complete an assessment questionnaire to help guide future development of the project. Originally, these questionnaires were in PDF format and could be downloaded from the project website. Students would download, print, then complete the questionnaires and hand them in with their experiment writeup. Students complained that this process was rather tedious and it could only be used for students at MIT. Another problem with this method was that the process of data collection and analysis was tedious since the survey results had to be entered into a computer before any analysis could take place. This process is highly infeasible if a large group of students were to be performing experiments and providing feedback on their experience.

The logical solution to this problem was to develop an online questionnaire that would store student feedback in a database where it could be easily retrieved and analyzed. The online questionnaire that was developed is shown in Figure 3-17. The student is asked to identify their team, if they belong to one, and for answers to each of the questions. Student responses to individual questions were rated on the scale of one to seven, with seven being the highest rating and one being the lowest. The questionnaire also contains comment fields for additional student input. Student comments have been extremely influential on the development of the system. In fact, the idea for an online questionnaire came from a student.

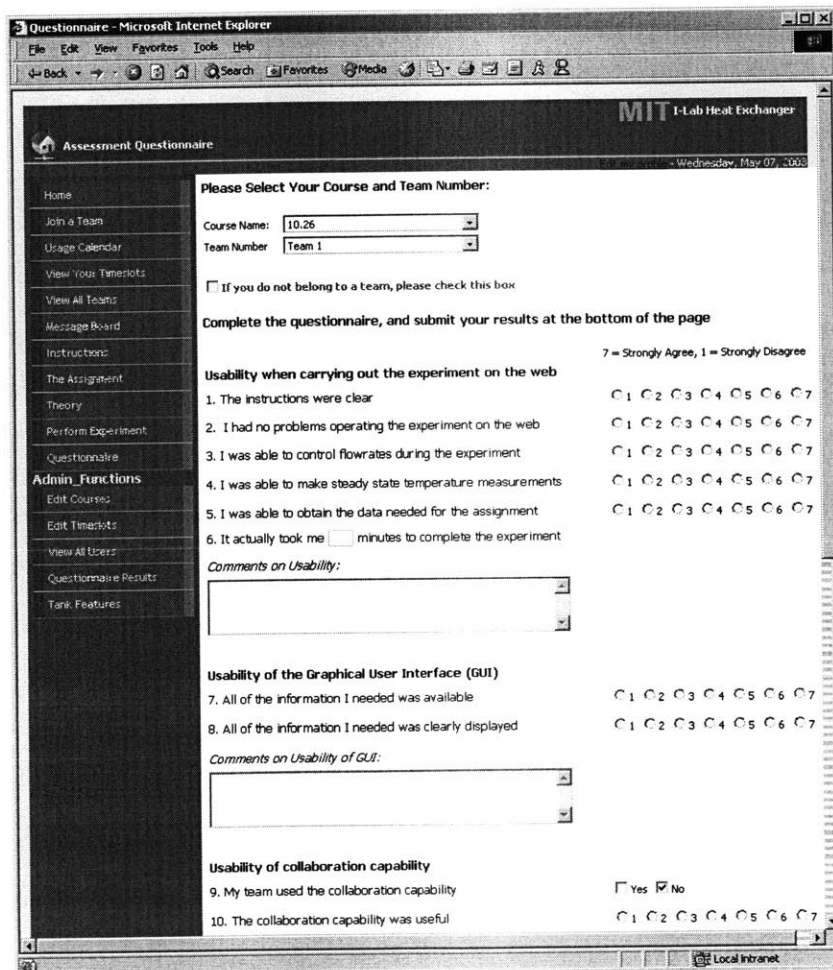


Figure 3-17: Assessment Questionnaire

The page also contains validator controls to ensure proper responses to questions that require numerical answers. For example, a validator control is used to ensure a student enters a value for the number of minutes it took to complete the experiment. The controls also check to see if the data entered is actually a number. When a student presses the “Submit Answers” button at the bottom of the page, a connection is opened with the SQL database, and the data is transferred from the web form and stored in the Questionnaire table. Later, the data can be viewed or can be exported into Excel format. This is discussed in the next section

### ***3.4.7 ADMINISTRATOR FEATURES***

As discussed earlier, administrator features on the website are only enabled when a particular user has administrator privileges. An administrator has the ability to edit the courses and teams that will be using the system, allocate available timeslots, give others administrator privileges, and view the results of the assessment questionnaires.

Figure 3-18 shows the questionnaire results page, where the administrator can view student responses to the assessment questionnaires. The student responses are contained in the table and can be exported into Microsoft Excel format by entering a filename with an .xls extension and clicking the “Export to Excel” button. The questionnaire data is saved locally on the server as an Excel file. Entering the correct web address in the browser will retrieve the saved data. The tabular data does not contain student comments about the experiment. Student comments can be viewed by clicking the “View Student Comments” link, which produces the page in Figure 3-19. Here the comments can be viewed and copied into a program such as Microsoft Word

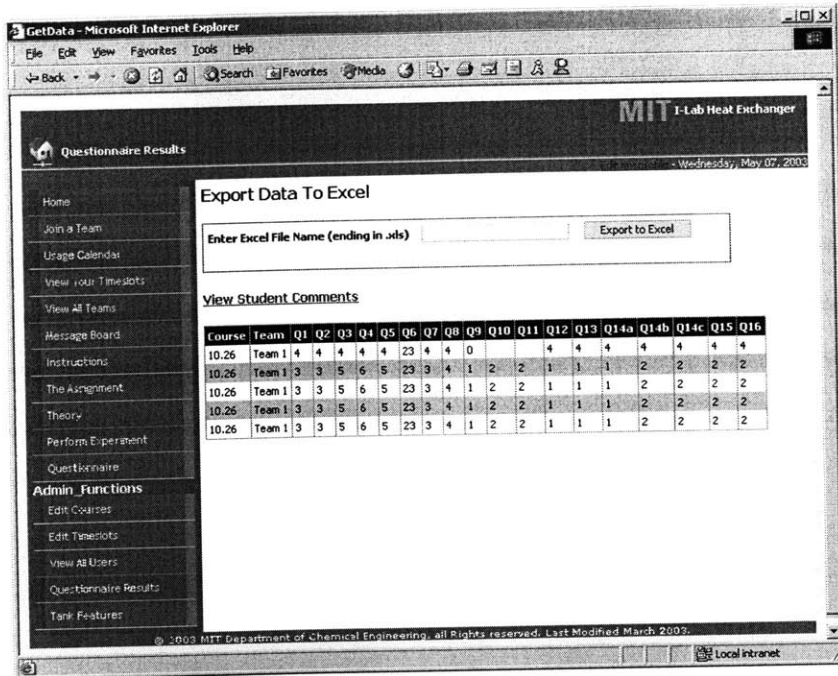


Figure 3-18: Questionnaire Results

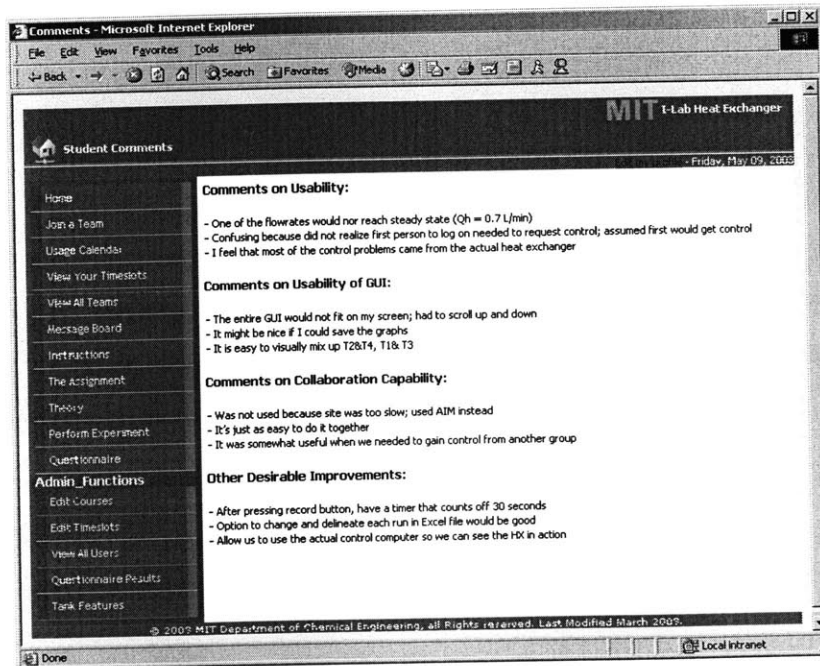
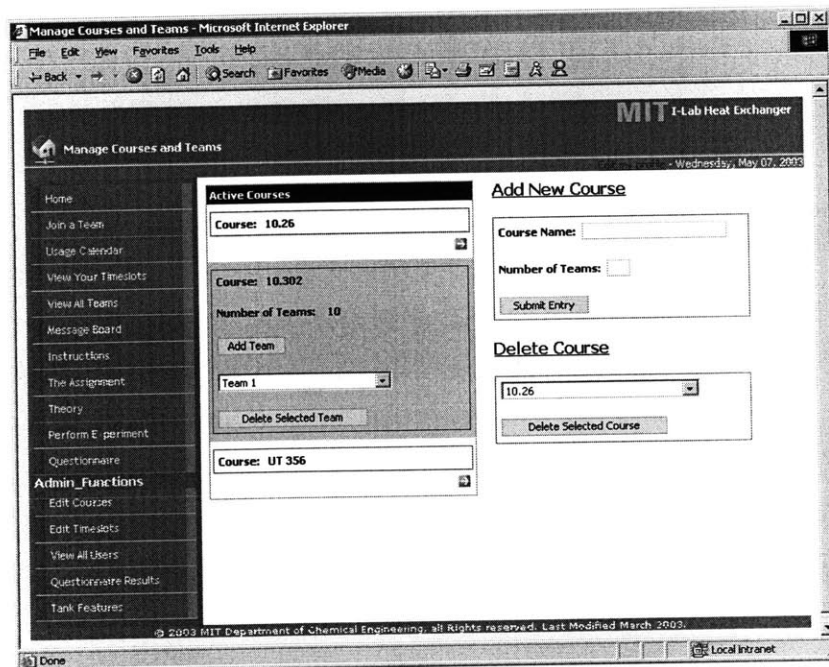


Figure 3-19: Student Comments

Figure 3-20 shows the course and team management page, through which course and team editing can be performed. On the left side of the page is a listing of all the courses

that are currently using and are recognized by the system, and on the right side are methods to add a new course or delete an existing one.

The Add Course dialog box takes as input a course name and the number of teams that are to be created for the course. Both input fields are validated through the use of a RequiredFieldValidator. Pressing “Submit Entry” sends the request to the SQL server, which then updates the Courses and CourseTeams tables. The new course becomes visible on the left hand side of the page. To delete a course from the system, an administrator can select the corresponding course from the dropdown list and click the “Delete Selected Course” button.



**Figure 3-20: Manage Courses and Teams**

On the left side of the page, labeled Active Courses, there are listings of all the courses that are currently have students performing experiments on the heat exchanger. By clicking on a blue arrow under a course name, an administrator can bring up the edit course panel shown in gray, allowing him/her the ability to add or delete teams from a particular course. By clicking the add team button, an administrator can add a team to a particular course, and by clicking the delete selected team button, an administrator will

delete the team that is currently selected from the dropdown list. If a team is chosen for deletion, all references to it will be removed from the database tables.

Figure 3-21 shows the edit timeslots page, through which an administrator can select the timeslots that will be available for students to perform experiments. The page is very similar to the calendar page in format, but it allows the administrator select the timeslots that will be available on each day. On the right are the available hours that the experiment could run. A check next to a time indicates that the time will be made available for student use. For any timeslot that is not checked, a “Not Available” will appear in the Occupied By field on the calendar page (Figure 3-14) preventing students from signing up for that particular timeslot.

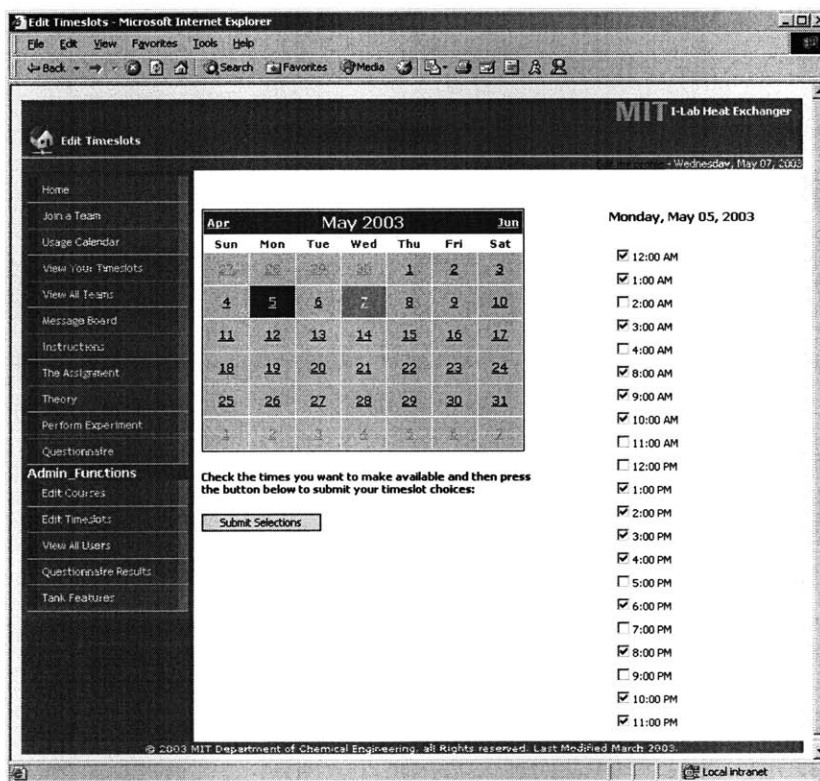


Figure 3-21: Edit Timeslots

An administrator also has the ability to view the names, contact information, and privileges of all the users who registered to use the system. This is shown in Figure 3-22.

To add additional system administrators, a current administrator must first click the blue arrow to view details about the user. Then the administrator can check the Administrator Privileges box and can click “Commit Change” to submit the value to the server. Upon successful submission, a confirmation message will be received.

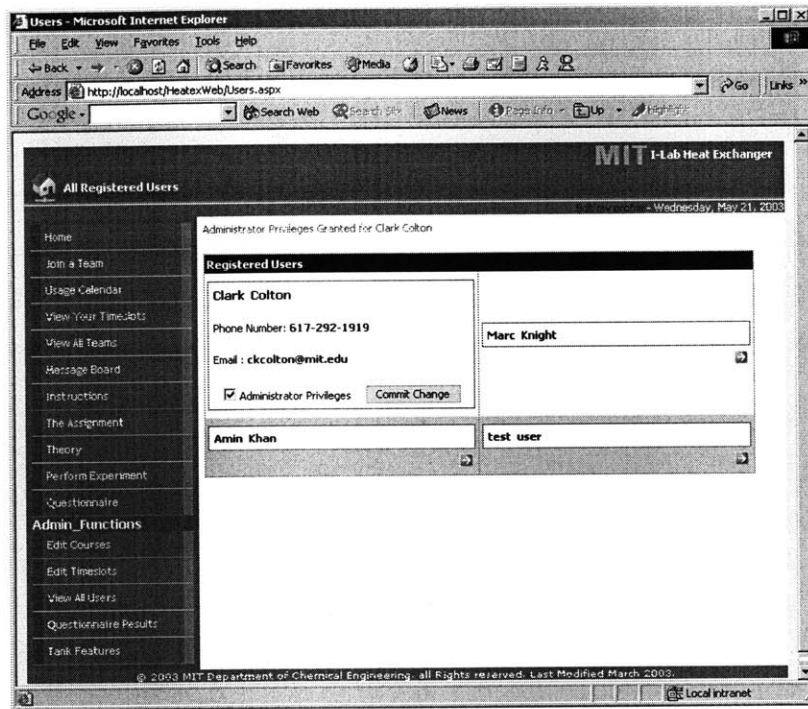
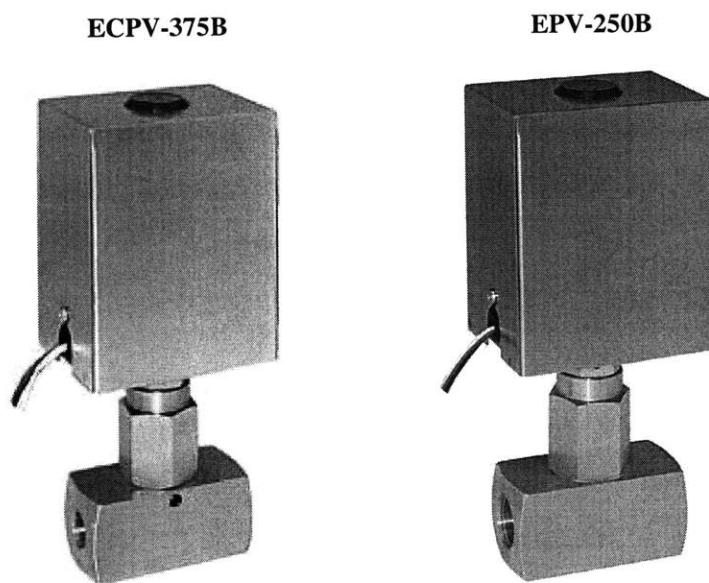


Figure 3-22: All Registered Users

### 3.5 REMOTE WATER RECYCLING

In order to make the experiment accessible 24 hours a day required that the system operate with minimum manual supervision. To ensure normal operation of the Heat Exchanger, the water in the Hot Water reservoir needs to be replaced daily to combat evaporation and fouling. The HT30XC service unit has no provisions for performing the water replacement automatically. If there was a way to perform the replacement automatically, the need to have a person responsible for maintenance of the reservoir could be eliminated. Automating the water replacement process was accomplished using two electronically controlled valves, a data acquisition board, and LabVIEW 6.1.

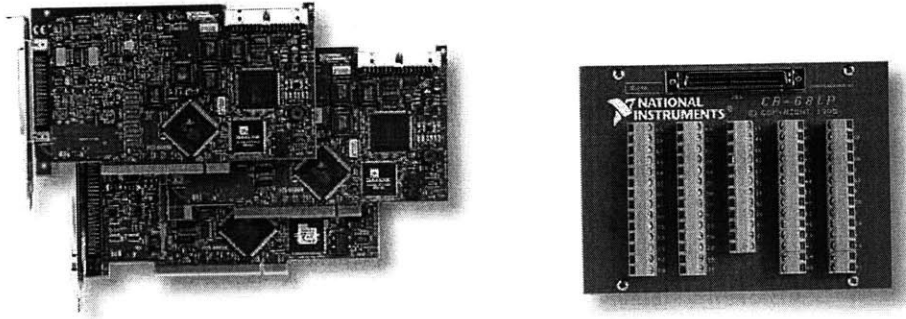
To control the flow of water into and out of the reservoir, two valves were used. The valves used are could be electronically controlled from a device such as a computer. Both valves were obtained from Hass Manufacturing Company, a leading manufacturer of electronically controlled valves. Two different types of valves were used. The EPV-250B Electronic Proportional Valve was used to control the flow of water into the reservoir and the ECPV-375B Electronically Controlled Pinch Valve was used to control the flow of water out of the reservoir. Both these valves are shown in Figure 3-23.



**Figure 3-23: Electronically Controlled Valves**

The valves accept a 4-20 mA or 1-5 VDC control signal. The control signal is processed by a 8-bit analog-to-digital converter contained in the valve housing. Additionally, a digital filter removes noise from the signal. Lastly, the microprocessor converts the control signal to a valve position and directs the 200 step per revolution DC step motor. In the pinch valve, this motor rotates a threaded stem whereas in the proportional valve the motor attaches to a zero-backlash screw mechanism. These mechanisms either pinch a tube or close a valve.

The National Instruments PCI-6024E low-cost data acquisition board and CB-68LP I/O Connector Block were used to supply the control signals to the valves. These parts are shown in Figure 3-24. The PCI-6024E board contains two 12-bit analog outputs; 8 digital I/O lines; and two 24-bit counters. Each of the valves was connected to an analog output channel of the data acquisition board that supplied the 1-5 VDC control signal.



**Figure 3-24: Data Acquisition Board & Connector Block**

LabVIEW 6.1 was used to generate the proper control signals for the valves. The LabVIEW interface allows the user to select one of two operations. The user can choose to empty the tank, or recycle the water in the tank. If the user chooses to empty the tank, the LabVIEW interface opens the pinch valve for a predetermined number of seconds, emptying the tank. If the user chooses to recycle the water, the pinch valve is opened for sixty seconds then closed after which the proportional valve is opened for seventy-five seconds then closes. This has the effect of emptying then refilling the tank with water. The LabVIEW interface is shown in Figure 3-25 and is accessible from the web making remote recycling of the water in the hot water reservoir possible.

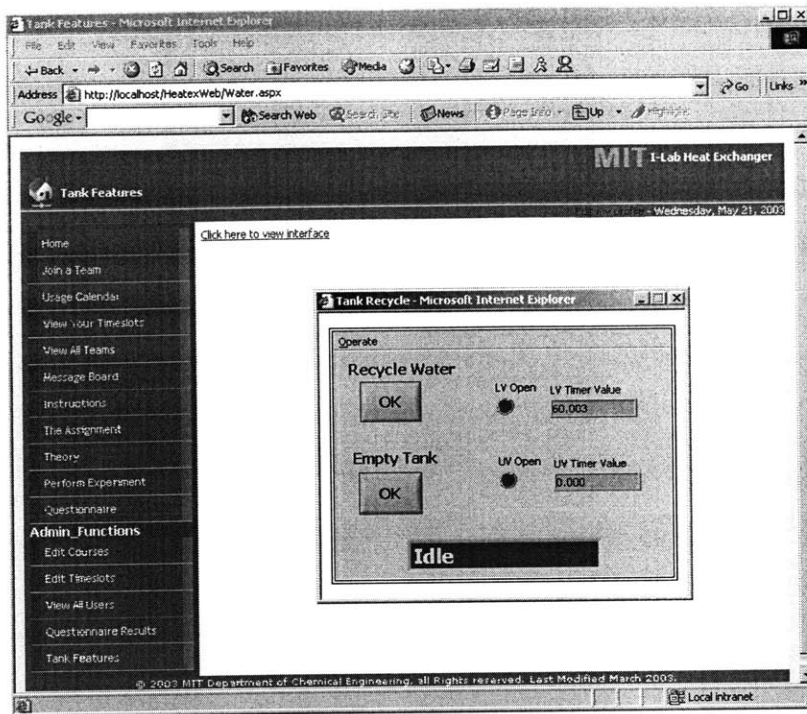


Figure 3-25: Interface for Remote Water Recycling

## 4 STUDENT FEEDBACK

One important driver behind development of the I-Lab Heat Exchanger has been student feedback. Since the project began two years ago, courses that issued I-Lab Heat Exchanger assignments had students complete assessment questionnaires. This feedback has been instrumental in determining what students' like/dislike about the weblab, problems they encountered, and improvements they would like to see made. A sample questionnaire is contained in the Appendix. The questionnaire contains questions divided into four broad categories:

- **Usability of the experiment:** Responses in this category helped us identify the problems associated with the system that need modification and improvement.
- **Improvements the students would like to see:** Responses in this category helped us design new features and integrate them to the system.

- **Meeting Educational Objectives:** Responses in this category helped us judge whether the Heat Exchanger Experiment met its educational goals
- **Overall Experience:** Responses in this category gave us a broad idea about how the students felt about the Heat Exchanger Experiment.

This academic year, we were able to obtain feedback on the I-Lab Heat Exchanger from two MIT courses 10.302 (Transport Processes - Fall 2002) and 10.26 (Chemical Engineering Projects Laboratory - Spring 2003).

## **4.1 10.302 (TRANSPORT PROCESSES) – FALL 2002**

10.302 provided an excellent opportunity to obtain feedback about the I-Lab Heat Exchanger. At the time, the collaboration utility and the LabVIEW 6.1 GUI had been developed, but remained untested by students. It was desirable to see if students enjoyed collaborating in teams for the assignment, or preferred the new LabVIEW interface when compared with the Java one.

### ***4.1.1 PART ONE – JAVA INTERFACE***

The 10.302 experiment was divided into two parts. For part one, students were assigned to teams of 3-4 people and were asked to take measurements operating the machine in countercurrent flow. Students completed the required analysis, and were asked to make predictions about what they would expect if they operated the machine in cocurrent flow. For this part, students used the Java Interface shown in Figure 3-3 to conduct the experiment.

Table 4-1 summarizes the results from the 40 students who completed the Part One questionnaires. It took a team of 3-4 students approximately 32 minutes to complete the assignment, which was a reduction over the amount of time it took previously.

(7 = Strongly Agree, 1 = Strongly Disagree)	Score ± St. Dev	Range	
		Min	Max
<b>Usability when carrying out the experiments on the web</b>			
1. The instructions were clear	6.0 ± 1.1	3	7
2. We had no problem operating the experiment on the web	5.9 ± 1.3	3	7
3. We were able to control flowrates during the experiment	6.2 ± 1.1	3	7
4. We were able to make steady state temperature measurements	5.8 ± 1.0	4	7
5. We were able to obtain the data needed for the assignment	6.4 ± 0.7	5	7
6. It actually took ____ minutes to carry out the experiment	32.1 ± 11.6	10	50
<b>Usability of the Graphical User Interface (GUI)</b>			
7. All of the information that I needed was available	6.4 ± 1.1	2	7
8. All of the information that I needed was clearly displayed	6.1 ± 0.9	4	7
9. The collaboration capability was useful	5.8 ± 1.3	3	7
<b>Improvements you would like to see on the website</b>			
10. Table of measurements vs. time	5.2 ± 1.6	1	7
11. Information on heating in the Hot Water Reservoir	4.9 ± 1.8	1	7
12. Basic equations of heat exchange	5.3 ± 1.6	1	7
<b>Meeting Educational Objectives</b>			
13. The remotely controlled experiment provided an experience of measurement of flowrates and temperatures in a real system analysis of real data consideration of measurement errors	5.8 ± 1.5	1	7
	6.2 ± 1.0	4	7
	5.8 ± 1.1	4	7
14. The assignment provided a vehicle for learning and reviewing thermodynamic limitations on performance, energy balance rate equations for heat exchange calculating overall heat transfer coefficients, correlations	5.8 ± 1.4	2	7
	5.8 ± 1.1	3	7
	5.7 ± 1.3	2	7
<b>Overall Experience</b>			
15. The I-Lab heat exchanger experiment was a beneficial experience	5.3 ± 1.3	3	7
16. The I-Lab heat exchanger experiment was fun	4.6 ± 1.7	1	7

**Table 4-1: 10.302 Part One Questionnaire Results**

Students had a lot of comments when asked about the experiment, GUI, and how to improve the system. Some students experienced problems downloading the plugin and controlling the parameters. Others found the system very easy to use, had no problems, and found it to be a valuable learning experience even though they thought the data analysis was tedious. Selected student comments are listed below:

**Comments on usability when carrying out experiment:**

- *“Operating experiment was relatively straightforward. It would have been nicer if we had actually seen the experiment setup rather than just the computer program.”*
- *“We had a hard time downloading the plug-in and getting it installed on a home computer and had to use Athena.”*
- *“The system was easy to use and control.”*

- "We got confused on how to actually access the data and whether in the instructions the 30 sec. meant real time or the time that the graph was giving us because they did not respond."
- "Took 35 minutes for me to log on due to inability to download Java plug-in on person PC and slow Athena computers."

**Comments on usability of GUI:**

- "We had problems with the interface freezing while we were trying to carry out the experiment."
- "Simple and easy to use."
- "More information could be given about the actual heat exchanger."
- "I think that the collaboration capabilities are good, but we chose to run it together and then work on the report."
- "Perhaps make the chat window scroll down automatically as new messages are typed."
- "Logging in twice was confusing."
- "Very good instructions!"

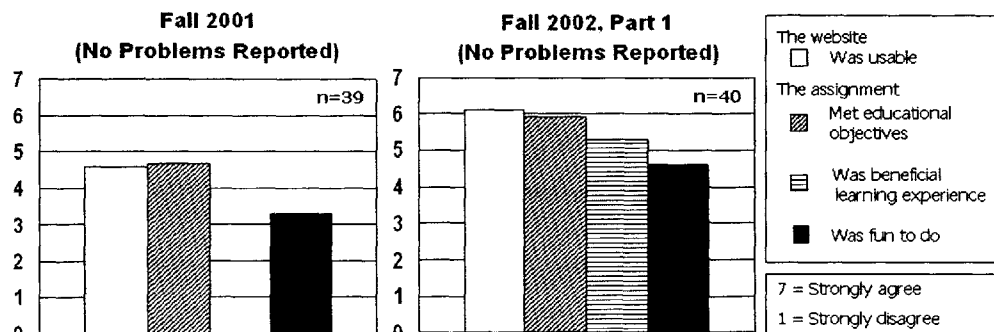
**Other desirable improvements:**

- Flow schematic
- More pictures of the heat exchanger to know what exactly is going on.
- Better compatibility with other computer platforms (i.e. all Athena computers).
- Feature that would allow mean temperature to be displayed.
- "The table would be helpful for evaluating if we had actually reached steady state or not because it could be hard to yell just by the graph."
- "Maybe a short tutorial about the various functions would be useful."

**Additional comments:**

- "Experiment went well overall!"
- "Without seeing heat exchanger in real life, it didn't feel any different than if we had been just given numbers."
- "It was good to be able to do this and see what a real heat exchanger is like but I would also like to have seen it in person as opposed to remotely."
- "Data analysis seemed a bit tedious. Questionnaire should be filled out online."

The questionnaires 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 4-1.



**Figure 4-1: Student Assessment Of I-Lab Heat Exchanger: Comparison Of 10.302 (Transfer Processes) In Fall 2001 And Fall 2002 Using The Java 2 Interface**

### 4.1.2 PART TWO – LABVIEW 6.1 INTERFACE

Table 4-2 summarizes the results of the 34 students that completed the Part Two Assessment Questionnaire. The main objective of the questionnaire was to find out if students preferred the newly developed LabVIEW interface to the Java interface used in Part One. When asked, 19 students preferred the LabVIEW interface compared with 15 who preferred the Java Interface. To get a better estimate of which interface was actually preferred, we decided to also analyze the subset of students who had problems controlling the equipment. This group preferred the Java interface and gave lower ratings when compared with whole class on almost every question. From the data, we concluded that the students liked both GUIs, but when asked to pick one they preferred the LabVIEW interface (when comparison is made between students who had no problems with either interface). The questionnaires 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. Another noticeable result from the questionnaire data was the fact that the collaboration capability received high scores even though some students indicated in their comments that they all worked at the same computer.

<i>(7 = Strongly Agree, 1 = Strongly Disagree)</i>	Problems	All Students
	Score ± St. Dev	Score ± St. Dev
<b>Usability when carrying out the experiments on the web</b>		
1. The instructions were clear	4.6 ± 1.4	5.5 ± 1.5
2. We had no problem operating the experiment on the web	2.1 ± 0.7	4.3 ± 2.2
3. We were able to control flowrates during the experiment	4.1 ± 1.6	5.6 ± 1.4
4. We were able to make steady state temperature measurements	3.4 ± 1.1	4.7 ± 1.7
5. We were able to obtain the data needed for the assignment	5.0 ± 1.7	5.6 ± 1.6
6. It actually took ____ minutes to carry out the experiment	72.1 ± 45.1	39.1 ± 31.4
<b>Usability of LabVIEW Graphical Use Interface (GUI) in Part 2</b>		
7. All of the information that I needed was available	6.0 ± 1.5	6.0 ± 1.2
8. All of the information that I needed was clearly displayed	5.7 ± 1.3	5.7 ± 1.5
<b>Comparison between GUI in Parts 1 and 2</b>		
9. I liked the Java 2 GUI in Part 1	6.0 ± 0.6	5.5 ± 1.2

10. I liked the LabVIEW GUI in Part 2	5.1 ± 1.5	5.3 ± 1.7
11. If I had to choose one, I prefer the:		
Java 2 GUI (Part 1)	5 Students	15 Students
LabVIEW GUI (Part 2)	2 Students	19 Students
<b>Usability of Collaboration capability</b>		
12. The collaboration capability was useful	6.0 ± 0.7	5.9 ± 1.5
13. The chat facility for communication was useful	6.6 ± 0.5	5.7 ± 1.7
14. I like the ability to easily switch control amongst team members	5.6 ± 0.5	5.9 ± 1.1

**Table 4-2: 10.302 Part Two Questionnaire Results**

Below are some selected student comments that illustrate different student preferences, problems encountered, and the limited usage of the collaboration capability.

**Comments on usability of GUI in Part 2:**

- *"Some functions did not work, large plug-in."*
- *"Too fancy, does not work well, more difficult to connect to."*
- *"More intuitive, "Enter" buttons to input the flowrates might help (in addition to the text boxes) but that may cause more programming annoyance."*
- *"More user-friendly than in Part 1."*
- *"The lab was quite confusing because we had so much trouble trying to get and maintain control of the system; once that was established, it was easier."*
- *"I liked it better than Part 1."*

**Comments on comparison between GUI in Parts 1 and 2:**

- *"For Windows users the LabVIEW interface is more familiar."*
- *"Part 1 part was simpler, more user-friendly, and more reliable. Part 2 was visually appealing, but more complicated/confusing and less consistent (sometimes "disappeared" for a few minutes)."*
- *"Part 2 interface is aesthetic, but not very user-friendly. Part 1 can be improved by highlighting the "Submit" button, otherwise all right."*
- *"I like the Part 1 GUI better because it is simpler and not crowded with excess colors and "gadgets"."*
- *"The data was more easily retrievable with the second part, whereas the 1<sup>st</sup> part's data was hard to work with on Athena."*
- *"LabVIEW is a wonderfully laid out GUI; with a few adaptations it would be even better."*
- *"I liked all the graphs and charts on the LabVIEW interface and the graph that showed we were at steady state."*

**Comments on collaboration capability:**

- *"This capability partially helped in getting a team member to install the correct LabVIEW plugin."*
- *"Good –real-time conversation was helpful."*
- *"Since we all worked on the same computer, collaboration was not necessary."*
- *"We did ours all at one computer so we didn't have to chat."*
- *"Very good."*
- *"I liked working in groups."*
- *"We all weren't on the same computer."*
- *"We only used 1 computer for data collection."*
- *"My group didn't use it."*
- *"We didn't use the collaboration capability although I think it would be useful."*

**Additional comments:**

- "The recording issues need to be debugged, but the LabVIEW controls are much easier to use than the Java GUI."
- "I liked the ability to access the system remotely (i.e. outside Athena/campus)."
- "Great if show data recorded in a different color on existing graph."

## 4.2 10.26 (CHEMICAL ENGINEERING PROJECTS LABORATORY) – SPRING 2003

A total of 46 students completed questionnaires after using the I-Lab Heat Exchanger in 10.26. Of this group, 93% of students used the I-Lab Heat Exchanger in 10.302. Since students preferred using the LabVIEW interface in 10.302, the LabVIEW interface was used again in 10.26. Table 4-3 summarizes the questionnaire results from the students in 10.26. The experiment received scores similar to those in 10.302, but this experiment took about an hour to complete as a result of taking more measurements. Not as many students reported problems with the equipment since they were already familiar with the equipment and it was likely that they already had the required plugins installed on their system. As has been the norm, students thought the experiment was a beneficial learning experience, but thought it wasn't fun to perform. Only one team out of 15 reported using the collaboration capability, which is surprising considering the comments of the 10.302 students. Students liked to meet as a group and perform the experiment around one computer instead of relying on the collaboration features.

(7 = Strongly Disagree, 1 = Strongly Agree)	Score ± St. Dev	Range	
		Min	Max
<b>Usability when carrying out the experiments on the web</b>			
1. The instructions were clear	5.8 ± 1.4	1	7
2. We had no problem operating the experiment on the web	4.7 ± 1.6	1	7
3. We were able to control flowrates during the experiment	5.0 ± 1.5	1	6
4. We were able to make steady state temperature measurements	5.4 ± 1.3	1	6
5. We were able to obtain the data needed for the assignment	5.9 ± 1.2	1	6
6. It actually took ____ minutes to carry out the experiment	60.0 ± 24.0	15	150
<b>Usability of LabVIEW Graphical User Interface (GUI)</b>			
7. All of the information that I needed was available	6.3 ± 1.3	1	6
8. All of the information that I needed was clearly displayed	2.0 ± 1.5	1	7
<b>Usability of collaboration capability</b>			
9. My team used the collaboration capability		Only 1 team out of 15 used this feature	

10. The collaboration capability was useful	4.3 ± 1.8	1	7
11. The chat facility for communication was useful	5.2 ± 2.0	1	7
<b>Improvements you would like to see on the website</b>			
12. Control over how data is averaged and reported	5.5 ± 1.6	1	7
13. Basic equations of heat exchange	5.1 ± 1.5	1	7
<b>Meeting Educational Objectives</b>			
14. The remotely controlled experiment provided an experience of measurement of flowrates and temperatures in a real system	6.0 ± 1.2	1	6
analysis of real data	6.1 ± 1.0	1	5
fitting data to a straight line	6.2 ± 0.9	1	4
preparing a schematic equipment diagram	5.5 ± 1.5	1	7
consideration of measurement errors	5.6 ± 1.2	1	5
15. The assignment provided a vehicle for learning how to write a technical report	5.5 ± 1.6	1	7
<b>Overall Experience</b>			
16. The I-Lab heat exchanger experiment was a beneficial experience	4.8 ± 1.7	1	7
17. The I-Lab heat exchanger experiment was fun	3.8 ± 1.7	1	7

**Table 4-3: 10.26 Questionnaire Results**

10.26 students did not have many comments, but some selected ones are shown below:

**Comments on Usability**

- *“One of the flowrates would not reach steady state ( $Q_h = 0.7 \text{ L/min}$ )”*
- *“Confusing because did not realize first person to log on needed to request control; assumed first would get control”*
- *“I feel that most of the control problems came from the actual heat exchanger”*

**Comments on Usability of LabVIEW GUI**

- *“The entire GUI would not fit on my screen; had to scroll up and down”*
- *“It might be nice if I could save the graphs”*
- *“It is easy to visually mix up  $T_2$  &  $T_4$ ,  $T_1$  &  $T_3$ ”*
- *“When the data was transferred to Excel, the temperature columns were mislabeled”*

**Comments on Collaboration Capability**

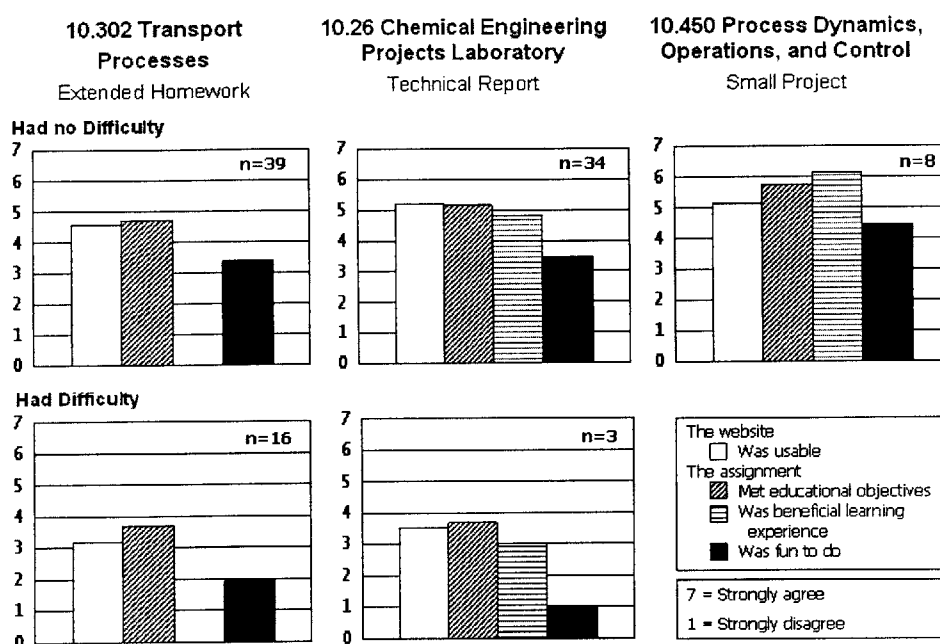
- *“Was not used because site was too slow; used AIM instead”*
- *“It’s just as easy to do it together”*
- *“It was somewhat useful when we needed to gain control from another group”*
- *“Rather than mess with the collaboration capability, we all worked at the same computer”*

**Other Desirable Improvements**

- *“After pressing record button, have a timer that counts off 30 seconds”*
- *“Option to change and delineate each run in Excel file would be good”*
- *“Allow us to use the actual control computer so we can see the HX in action”*

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.



**Figure 4-2: Student Assessment of I-Lab Heat Exchanger: Comparison of Evaluations in Different Classes**

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.

### **4.3 NOTABLE STUDENT SUGGESTIONS**

As a result of the Questionnaires, there are many ways that the I-Lab Heat Exchanger Project can continue to be improved. Some of the previous suggestions dealt with the addition of a WebCam, improvements to the chat features, and an online questionnaire. At this time, all these suggestions have been implemented. The WebCam is integrated into the default start page of the website to provide a realtime look at the equipment. Ease of use suggestions such as word-wrapping have been integrated into the chat window, making it more similar to popular chat programs such as AOL Instant Messenger. Finally, the online questionnaire reduces the need for traditional paper questionnaires and tedious data collection measures. These suggestions have greatly improved the overall experience of performing an experiment on the I-Lab Heat Exchanger.

## **5 CONCLUSION AND FUTURE IMPROVEMENTS**

During the course of work on this thesis, many significant improvements were made to the I-Lab Heat Exchanger Project. Setting clear goals and incorporating feedback from students allowed these improvements to be implemented over a one-year period. As more experiments are performed on the I-Lab Heat Exchanger during the upcoming terms, various improvements will continue to be made to the system.

### **5.1 SUMMARY OF ACCOMPLISHMENTS**

Over the past year, there have been five key accomplishments with regards to I-Lab Heat Exchanger system development. These accomplishments are summarized below:

- Developed collaboration utility to facilitate team-based use of the I-Lab Heat Exchanger. The utility featured a chat window, user list, and a “Pass the Token” capability to determine which user could control the experiment parameters. A collaboration management server was also developed to handle message passing and coordination between multiple clients.
- Developed a new client interface using LabVIEW 6.1. This interface was made web-accessible using features specific to this release of LabVIEW. The interface also featured a more refined graphical display to help the student understand what was taking place at the heat exchanger.
- Found optimal PID parameters for hot water control algorithm. This enabled students to have accurate control of the temperature in the hot water reservoir regardless of heat exchanger type, flow arrangement, or flow rate.
- Developed a user management system for the I-Lab Heat Exchanger website. The system featured a WebCam and functionality for users to join teams and sign up for timeslots. Furthermore, it contained a message board and special administrator features.
- Developed infrastructure to allow for the recycling of the fluid in the hot water reservoir. This was accomplished using electronically controlled valves, a data acquisition board, and the LabVIEW 6.1 software.

## 5.2 FUTURE WORK

Looking toward the future, there are several improvements that can be made to the I-Lab Heat Exchanger at this time.

- Currently, the heat exchanger’s cold water stream is tied directly into the building tap water supply. This makes the cold inlet temperature dependent on the tap water temperature which could fluctuate based on season, time of day, etc... This fact makes it hard to repeat an experiment with exactly the same temperatures. A solution to this problem would be to integrate a chiller unit into the system that would keep the

cold water inlet temperature constant. It would function similarly to the hot water reservoir, circulating water through the system using a computer controlled pump.

- In the future, there may be instances where a group of students may wish to perform an experiment using different types of client interfaces. For example, one student may prefer the LabView 6.1 interface, and another might not want to download the plugin and would rather use the Java interface. Although the current architecture supports both interfaces, it lacks a mechanism to communicate information between the two. Currently, no mechanism exists to restrict control of the experiment parameters when students are concurrently running the LabView 6.1 interface as well as the Java one. There is a “Pass the Token” capability built into LabView 6.1 that restricts control of the parameters to one user, but there is no “bridge” to integrate this capability with the collaboration management server and Java interface. If there was a way to integrate both systems, concurrent use of the LabView 6.1 and Java interfaces could be offered.
- Several pages that compose the User Management system are not optimized for viewing in Netscape Navigator or the Mozilla browsers. Changes in the HTML code may result in a more consistent page appearance across browsers.
- Refilling the hot water reservoir is not always accurate since a timer is used to determine how long the valve stays open. The amount of water that ends up in the tank is subject to pressure variation in the tap water line. Integrating a conductivity level sensor into the system would provide the proper feedback to eliminate the timer and ensure that the tank refills to the same level each time, regardless of pressure.
- It would also be desirable to integrate a calendar into the interface that recycles the water in the hot water reservoir. This would enable automatic recycling at an interval decided by the administrator. For example, the water could be recycled twice a day, daily, or every two days.

The I-Lab Heat Exchanger Project is an experiment that can continue to be developed to meet the needs of students at MIT and at other universities. The features introduced in this thesis are designed to be robust and can easily adapt to change. Student response to the experiment has been favorable in general and is likely to improve as more features are implemented. Hopefully the success of the I-Lab Heat Exchanger Project will foster development of other remotely controlled laboratory experiments in chemical engineering at this university and at others.

# APPENDIX

## Assessment Questionnaire 10.26 I-Lab Heat Exchanger Assignment, Spring 2003

Name:

Year:

I did the I-Lab Heat Exchanger Assignment in 10.302. \_\_\_ Yes \_\_\_ No

Score

1 = Strongly Agree

7 = Strongly Disagree

If yes, please comment on any improvements you noticed.

### Usability when carrying out the experiment on the web

1. The instructions were clear. \_\_\_\_\_
2. I had no problems operating the experiment on the web. \_\_\_\_\_
3. I was able to control flowrates during the experiment. \_\_\_\_\_
4. I was able to make steady state temperature measurements. \_\_\_\_\_
5. I was able to obtain the data needed for the assignment. \_\_\_\_\_
6. It actually took me about \_\_\_\_\_ minutes to carry out the experiment. \_\_\_\_\_

Comments on Usability:

### Improvements you would like to see on the website

7. Control over how data is averaged and reported \_\_\_\_\_
8. Basic equations of heat exchange \_\_\_\_\_

Other desirable Improvements:

### Meeting Educational Objectives

9. The remotely controlled experiment provided an experience of  
measurement of flowrates and temperatures in a real system \_\_\_\_\_  
analysis of real data \_\_\_\_\_  
fitting data to a straight line \_\_\_\_\_  
preparing a schematic equipment diagram \_\_\_\_\_  
consideration of measurement errors \_\_\_\_\_
10. The assignment provided a vehicle for learning how to write a  
technical report \_\_\_\_\_

### Overall Experience

11. The I-Lab heat exchanger experiment was a beneficial learning experience. \_\_\_\_\_
12. The I-Lab heat exchanger experiment was fun. \_\_\_\_\_

Put any additional comments on the back including any benefits you derived from the experiment.

## REFERENCES

- [1] America Online Website. <<http://www.aol.com>>, April 2003.
- [2] Armfield Limited, “*Instruction Manual: Computer Controlled Heat Exchanger Service Unit HT30XC (HT30X-Remote-AL11036)*”, July 2001.
- [3] Bartlett, Dean A., “*The Fundamentals of Heat Exchangers*”, American Institute of Physics, 1996, <http://www.aip.org/tip/INPHFA/vol-2/iss-4/p18.pdf>
- [4] Khan, R, A “*Software Architecture for Web-Accessible Heat Exchanger Experiment*”, SM Thesis, Department of Civil and Environment Engineering, MIT, 2002.
- [5] Microsoft, “*SQL server 2000 for JDBC, User’s Guide and Referece*”, April 2002.
- [6] National Instruments, “*The PID Control Toolkit for G Reference Manual*”, National Instruments Corporation, Austin, Texas.
- [7] National Instruments, “*Integrating the Internet into Your Measurement System: DataSocket Technical Overview*”, National Instruments White Paper No. 1680, National Instruments Corporations, Austin, TX.