Web and Java Based Architecture for Laboratory Experiments

By

Mustafa Özgür TUTUM

A Dissertation Submitted to the Graduate School in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

Department: Computer Engineering

Major: Computer Software

Izmir Institute of Technology

Izmir,Turkey

June, 2003

We approve the thesis of **Mustafa Özgür Tutum**

Date of Signature

---------- ----------

Prof. Dr. Sıtkı AYTAÇ

Supervisor

Department of Computer Engineering

---------- ----------

Assoc. Prof. Dr. Ahmet KOLTUKSUZ

Department of Computer Engineering

---------- ----------

Assist. Prof. Dr. Şevket GÜMÜŞTEKiN

Department of Electrical and Electronics Engineering

---------- ----------

Prof. Dr. Sıtkı AYTAÇ

Head of Department

ACKNOWLEDGEMENTS

The author would like to express his gratitude to his thesis adviser, Prof. Sıtkı AYTAÇ, Ph.D, for his encouragement, guidance and support throught the development of this thesis.

The author would like to thank all friends and colleagues at the department for their support.

The author would especially like thank to his friend, Burak Galip ASLAN for his synergy .

The author would also like to express his gratitude to his parents for their patience.

ÖZETÇE

İletişim teknolojilerindeki gelişmeler eğitim faaliyetlerinin iletişim kanallarını kullanmasını yaygınlaştırmıştır. Bu kanallardan biri olan İnternet, laboratuar uygulaması gerektirmeyen eğitimlerin vazgeçilmez bir parçası olmuştur. Eğitmen ve öğrenciler coğrafi konumdan bağımsız olarak İnternet ortamında buluşmaktadır. Ne var ki farklı olan ihtiyaçlar ve bu ihtiyaçları karşılamaktaki güçlükler laboratuar uygulaması gerektiren eğitimlerin İnternet üzerinde aynı ölçüde yaygınlaşmasını engellemektedir. Bu çalışmada laboratuvar uygulamalarının İnternet üzerinden gerçekleştirilmesinde kullanılabilecek metodolojiler tartışılmaktadır. İhtiyaçlar yeniden gözden geçirilerek İYTE' deki uygulamalar için rafine edilmiştir. Saptanan ihtiyaçları karşılayacak bir mimari önerilmiştir ve bu mimari üzerinde geliştirilebilecek bir yazılımın analizi yapılmıştır.

ABSTRACT

Developments in the communication technologies area have increased the popularity of usage of communication channels for the education activities. Being one of these communication channels, Internet, has become an inevitable component for theoretical education. Internet has brought together instructor and student without any geographical constraint. But having different requirements and difficulties during the satisfaction of these requirements are the common drawbacks for the widespread usage of Internet for laboratory educations. In this thesis, methodologies, which can be used in remote laboratory systems, have been discussed. Requirements have been scrutinized and refined. Remote laboratory architecture for IYTE has been proposed according to these requirements. Analysis of software that can be developed on the proposed architecture has been made.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter1
MOTIVATION

1.1 Remote Experiment Concept

Education process is composed of three basic components. These are education source, student and relationship between this source and student. Written documents, audiovisual presentations and instructor can be an education source. Relationship between these sources and student is called as an education method. In education, major aim should be to increase the student's reasoning skills.

Reasoning can be performed by using induction, deduction and retroduction methods in a complex order. Deduction is the mental process of forming conclusions based on premises. The conclusions must follow directly and necessarily from the premises. Retroduction is a pattern of inference, which accounts for the hypothetical causes of unobservable entities as verifiable, meaningful objects of a scientific inquiry. And finally; induction can be described as collecting the individual bits of verifiable information that exist around us and then trying to arrive at general truths. In order to collect the necessary information for induction, observation or experiment should be done.

Observation is collection process of information. Observation must have two properties in order to be scientifically meaningful. One property is reliability. For reliable observation observer should be objective and observation should have a tolerable error. Other property is validity. Collecting information without any aim is not a valid observation.

Another method for collecting the information is doing an experiment. While observer does not change the facts in observation, in experiment scientist plays with the facts. Scientist makes systematic changes in initial conditions of the observation and observes the dependent conditions. Here, initial conditions are independent variables. In order to make systematic changes and necessary controls, an artificial observation environment is needed. Laboratories provide this environment.

Being two basic components laboratory equipments and student determine the laboratory cost. Cost is depending on expensive equipments and students' physical presence in the lab. Remote experimentation solution reduces the laboratory cost. Expensive laboratory equipments should be shared among students. Hence, there is no need for physical presence of students.

1.2 Overview of the Thesis

This chapter will continue with related work sub title. At that part, recent advances in remote experimentation subject are introduced.

In the second chapter, requirements of remote experimentation such as computer controlled experiment setup and multimedia support is discussed.

In the third chapter, feasibility study that has been done for IYTE is presented. Emulator solution is proposed as the first step of the real remote experimentation. Mathematical model of emulator and implementation details are explained.

Fourth chapter covers the object oriented analysis and design of proposed system . Unified Modelling Language is used for analysis and design.

Last chapter gives the conclusion and proposes the future extensions of this work.

1.3 Related Research and Recent Advances

Many institutions of higher education have successfully established Web-based environments where learners can pursue their higher education via Internet or alternative distance education methods. The success, however, has been concentrated on the domain of virtual classroom (Hirumi&Bermudez, 1996). But, still many learners whose courses are heavily lab-dependent are not able to enjoy the multidimensional benefits of real laboratory experimentation via the virtual classroom due to its technical limitations (Aktan, Bohus, Crowl, Shorl, 1996). In distance teaching, laboratory experimentation is inconvenient because the students usually have to be physically present in the universities' labs. One solution to avoid this disadvantage is virtual experimentation. In this paradigm the

experiments are simulated and visualized by means of virtual reality (Schimid, 1999). Another concept is remote control of laboratory experiments (Alhalabi, Anandapuram & Hamza, 1998).

Currently, various studies are being executed about remote laboratory subject, but two of them are much more important than the others.

The University of Western Australia executes one of these projects. They state that over $2 million had been invested for this system since 1993. Their telerobot research team has developed reliable Internet software and user interfaces which can be tailored for remote control and operation of any kind of equipment. And their web-based teaching team has shown how students can effectively learn basic theoretical material through computers and how staff can track their progress. They are trying to combine these two studies to provide remote access to teach laboratory equipment. Their system allows student to operate the equipment and collect data automatically. Labview program is at the core of the system. Labview is commercial software for instrumentation of experiments. Software includes drivers for various type of PC interface card (RATL, 2003).

Another important project continues at Federal University of Santa Carina in Brazil. Their system gives students an opportunity to run an assembly code remotely for 8051 microcontroller. Their lab is composed of a board containing 8051 microcontroller and other devices for the communication between the microcontroller and the server. They have server software that receives information from the client. Then software takes this information to the microcontroller and the results back to the client. Web browser is not enough for client in this system so they use client side software that loads student's machine code and sends it to the server. Here no visual feedback is used. It can be asked that what is the difference from the simulation of the microcontroller, but here students use real equipments and when an unexpected situation occurs, system responses real results (RELB, 2003).

Also, there are some other projects, but these two have partially finished and they are leading projects in this area.

Chapter 2

REQUIREMENTS OF REMOTE EXPERIMENTATION

2.1 Introduction

In engineering education, experiments play an important role. Experiment setups can be described as the light version of the real and complex engineering problems. Therefore, we should carefully determine which experiment could be done and which couldn't be through Internet.

Clearly we can state that computer based controller implementation is necessary for remote experimentation. With this statement we have changed our problem domain. Henceforth we will only deal with controller, regardless the type of experimental setup.

Other component of the remote experiment system is scheduling. Experiment setup must be shared among students. Registered students select the time that they will do their experiment. These processes should be executed by the access control mechanism.

Another important component is multimedia support. Visual feedback should be supplied to the student. With the visual feedback, the student supervises the experiment and checks whether the performance of the process is as expected.

Therefore we will examine if Java software development environment is enough or not to meet the needs belonging to three basic components of a remote experiment system. Controller design, access management and multimedia support… The reason that we examine Java is only for research purposes. It may be interesting to see that all-Java solution is not feasible.

In the next part of the chapter we will concentrate on these three basic components.

2.2 Controller Design

In the previous part we have stated that computer based controller implementation is necessary for remote experimentation. Therefore, necessary definitions about control systems will be given, before we explore the controller alternatives appropriate for remote experiments. Then, common control actions will be introduced.

2.2.1 Basic Definitions Related to Control Systems

2.2.1.1 Plant

A plant is a piece of equipment, perhaps just a set of machine parts functioning together, the purpose of which is to perform a particular operation (Ogata, 1990). In this thesis we shall call any physical experiment object to be controlled a plant.

2.2.1.2 System

A system is a combination of components that act together and perform a certain objective (Ogata, 1990).

2.2.1.3 Disturbance

A disturbance is a signal that tends to adversely affect the value of the output of a system. If a disturbance is generated within the system, it is called internal, while an external disturbance is generated outside the system and is an input (Ogata, 1990).

2.2.1.4 Feedback Control System

A system that maintains a prescribed relationship between the output and some reference input by comparing them and using the difference as a means of control is called a feedback control system (Ogata, 1990).

2.2.1.5 Closed-Loop Control Systems

Feedback control systems are often referred to as closed-loop control systems. In a closed-loop control system the actuating error signal, which is the difference between the input signal and the feedback signal, is fed to the controller so as to reduce the error and bring the output of the system to a desired value (Ogata, 1990).

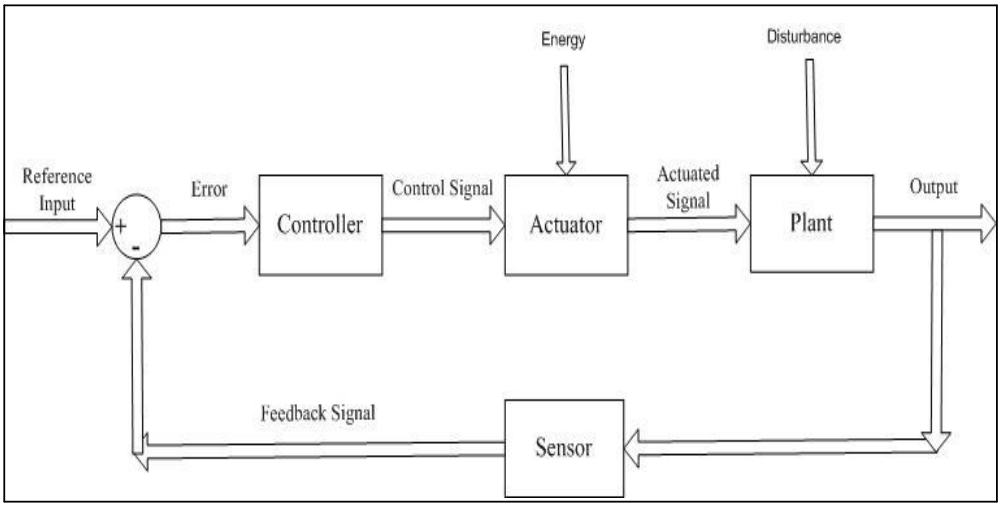

Figure 2.1. A typical closed-loop control system

2.2.1.6 Open-Loop Control Systems

Those systems in which the output has no effect on the control action are called open-loop control systems. In other words, in an open-loop control system the output is neither measured nor fed back for comparison with the input. Open-loop control can be used, in practice, only if the relationship between the input and output is known (Ogata, 1990).

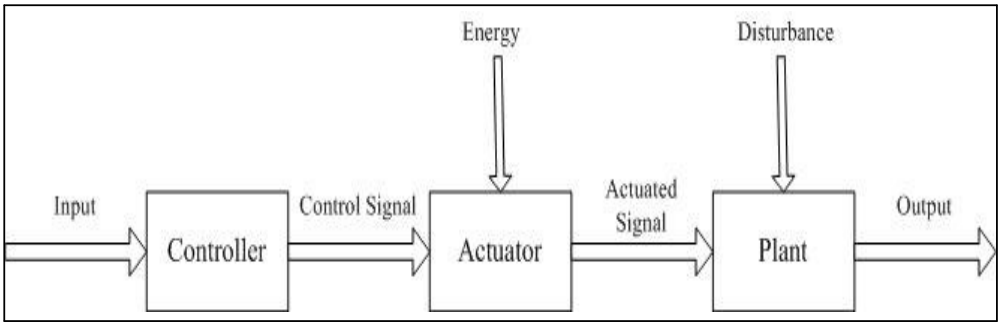

Figure 2.2. An Open-loop control system

2.2.1.7 Actuator

Actuator is a power device that produces the input to the plant according to the control signal so that the feedback signal will correspond to the reference input signal. The output of controller is fed to an actuator, such as a pneumatic motor or valve, a hydraulic motor, or an electric motor (Ogata, 1990).

2.2.1.8 Sensor

The sensor or measuring element is a device that converts the output variable into another suitable variable, such as a displacement, pressure, or voltage, that can be used to compare the output and the reference input signal (Ogata, 1990).

2.2.2 Common Control Actions

There are six basic control actions are very common among industrial controllers: on- off, proportional, integral, proportional-plus-integral, proportional-plus-derivative and proportional-plus-integral-plus-derivative control action. Before examining these control action we need to explain two concepts Laplace transform and transfer function.

The Laplace transform of function f(t) is given by

$$\ell [f(t)] = F(s) = \int_0^{\infty} f(t)e^{-st}\, dt \qquad \text{(Kuo, 1991)}$$

(eq. 2.1)

The transfer function of a linear, time-invariant, differential equation system is defined as the ratio of the Laplace transform of the output to the Laplace transform of the input under the assumption that all initial conditions are zero (Kuo, 1991).

The transfer function of a system is a mathematical model. It is an operational method of expressing the differential equation that relates the output variable to the input variable. A transfer function gives a full description of the dynamic characteristics of the linear and time-invariant system, as distinct from its physical description (Ogata, 1990).

2.2.2.1 On-off Control Action

Actuating element has only two fixed positions. If the output signal from the controller is u(t) and the actuating error signal is e(t), the signal u(t) remains at either a maximum or minimum value, depending on whether the actuating error signal is positive or negative (Ogata, 1990).

$u(t) = U_1$ for $e(t) > 0$ (eq. 2.2)

$u(t) = U_2$ for $e(t) < 0$ (eq. 2.3)

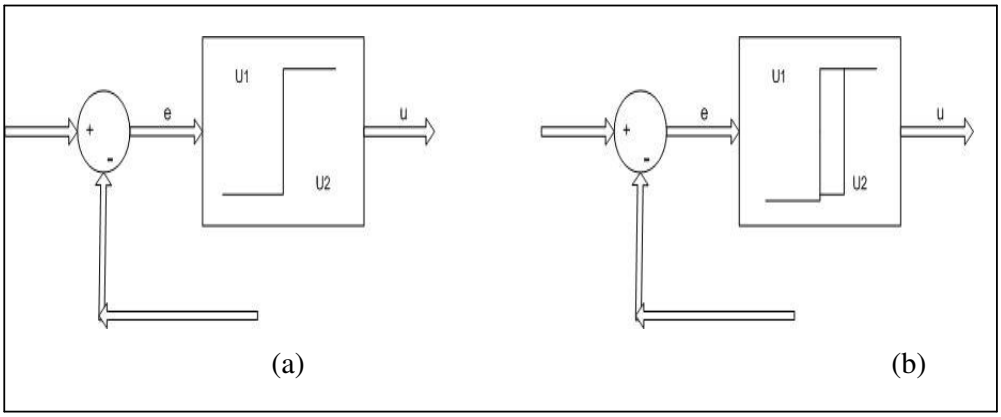

Figure 2.3. (a) Block diagram of an on-off controller (b) Block diagram of an on-off controller with differential gap

A differential gap is indicated Figure 2.3 (b). This gap causes the controller output u(t) to maintain its present value until the actuating error signal has moved slightly beyond the zero value. It may be the result of unintentional

friction. However, it is intentionally provided in order to prevent too frequent operation of the on-off mechanism (Ogata, 1990).

2.2.2.2 Proportional Control Action

The relationship between the output of the controller u(t) and the actuating error signal e(t) is

$$u(t)=K_p*e(t) \qquad \text{(eq. 2.4)}$$

, where K_p is the proportional gain. In Laplace transform quantities,

$$T(s)=U(s) / E(s) = K_p \qquad \text{(eq. 2.5)}$$

, where T(s) is the transfer function of the controller (Ogata, 1990).

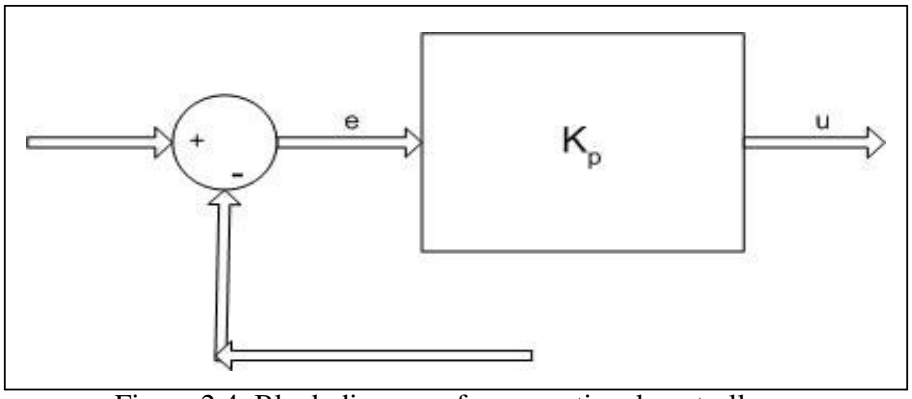

Figure 2.4. Block diagram of a proportional controller

2.2.2.3 Integral Control Action

The value of the controller output u(t) is changed at a rate proportional to the actuating error signal e(t). It is

$$u(t)=K_I \int_0^t e(t)\, dt \qquad \text{(eq. 2.6)}$$

, where K_I is an adjustable constant.

The transfer function of the integral controller is

$$T(s)=U(s) / E(s)=K_I / s \qquad \text{(eq. 2.7)}$$

In the proportional control of a plant whose transfer function does not possess an integrator 1/s, there is a steady-state error in the response of a unit step input. Such error can be eliminated if the integral control action is included in the controller (Ogata, 1990).

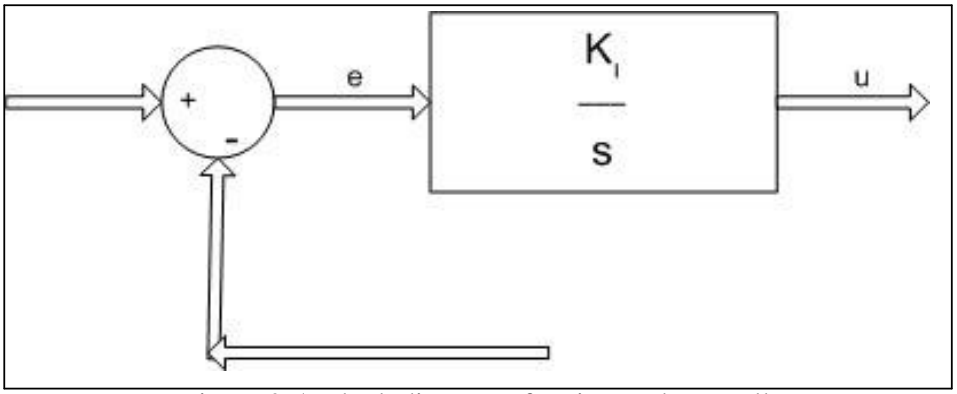

Figure 2.5. Block diagram of an integral controller

2.2.2.4 Proportional-Integral Control Action

The control action is defined as:

$$u(t) = K_p * e(t) + K_p / T_I \int_0^t e(t)\, dt \qquad \text{(eq. 2.8)}$$

, where T_I is the integral time.

The transfer function T(s) is

$U(s) / E(s) = K_p * (1 + 1/ (T_I * s))$ (Ogata, 1990) (eq. 2.9)

2.2.2.5 Proportional-Derivative Control Action

The control action is defined as:

$u(t) = K_p * e(t) + K_p * T_d * de(t)/dt$ (eq. 2.10)

The transfer function T(s) is

$U(s) / E(s) = K_p * (1 + T_d * s)$ (Ogata, 1990) (eq.2.11)

Derivative control action provides high sensitivity. An advantage of using derivative control action is that it responds to the rate of change of the actuating error and can produce a significant correction before the magnitude of the actuating error becomes too large. Although derivative does not affect the steady-state error directly, it adds damping to the system and thus permits the use of a larger value of the gain K, which will result in an improvement in the steady-state accuracy. Because derivative control operates on the rate of change of the

actuating error and not the actuating error itself, this mode is never used alone (Ogata, 1990).

2.2.2.6 Proportional-Integral-Derivative Control Action (PID)

It is the combination of proportional, integral and derivative control actions. The equation of a controller with this combined action is given by

$$u(t) = K * e(t) + (K_p/T_I) * \int_0^t e(t)\,dt + K_p T_d * (de(t)/dt) \qquad \text{(eq.2.12)}$$

or the transfer function is

$$T(s) = U(s) / E(s) = K_p * [1 + (1 / (T_i * s)) + (T_d * s)] \qquad \text{(eq.2.13)}$$

, where K_p is the proportional gain, T_i is the integral time and T_d is the derivative time. They can be called as P, I and D parameters of the controller. In summary, it has three parameters and these parameters describe the model (Ogata, 1990).

PID model is frequently used in industry. It is simple and powerful model. Thus we will use this model in remote controller design. In our design we will permit student to give an appropriate combination of three control parameters, K_p, T_i and T_d through the Internet and to perform the experiment .

2.2.3 Mathematical Model of Remote Experiment Plant

Although we concentrate on the design of the controller, we should define the model of our experiment setup or plant in control terminology. Most of basic real plants have no more than third order differential equation definition. Also controller solutions for second order systems are similar for higher order systems. Thus, we can design our controller for second order systems.

Second order systems are the systems that can be defined with second order differential equations. Transfer function of the system describes its dynamic behavior.

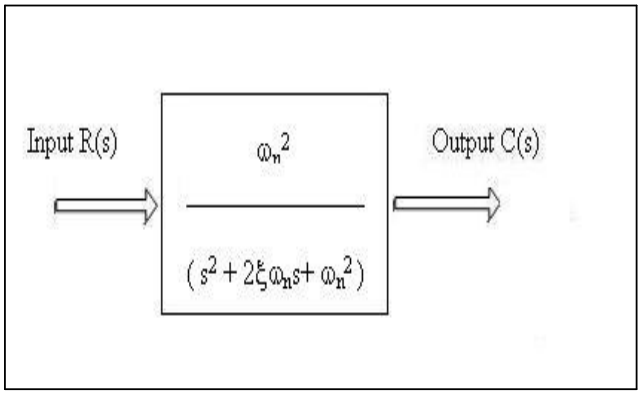

Figure 2.6. Block diagram of second order system

$T(s) = C(s) / R(s) = \omega_n^2 / (s^2 + 2\xi\omega_n s + \omega_n^2)$ (Sarıoğlu, 1999) (eq.2.14)

Also the dynamic behavior of the second order system can be described in terms of two parameters ξ and ω_n. ω_n is the undamped natural frequency and ξ is the damping ratio of the system. If $0 < \xi < 1$, then the system is called underdamped and the transient response is oscillatory. If $\xi = 1$, the system is called critically damped. Overdamped systems corresponds to $\xi > 1$. The transient responses of critically damped and overdamped systems do not oscillate. If $\xi = 0$, the transient response does not die out (Sarıoğlu, 1999).

2.2.4 Transient Response Characteristics of the Plant

2.2.4.1 Delay Time (td)

The delay time is the time required for the response to reach half of the final value at the first time (Ogata, 1990).

2.2.4.2 Rise Time (tr)

The rise time is the time required for the response to rise from 0% to 100% of its final value (Ogata, 1990).

2.2.4.3 Peak Time (tp)

The peak time is the time required for the response to reach the first peak of the overshoot (Ogata, 1990).

2.2.4.4 Maximum Overshoot (Mp)

The maximum overshoot is the maximum peak value of the response curve measured from the final steady-state value (Ogata, 1990).

2.2.4.5 Settling Time (ts)

The settling time is the time required for the response curve to reach and stay within a range about the final value. This range changes between 2% and 5% of the final value (Ogata, 1990).

These specifications are quite important since if we specify the values of these, then the shape of the response is virtually determined. Another important note is the conflict between the maximum overshoot and rise time. Both of them cannot be made smaller simultaneously. If one of them is made smaller, the other becomes larger.

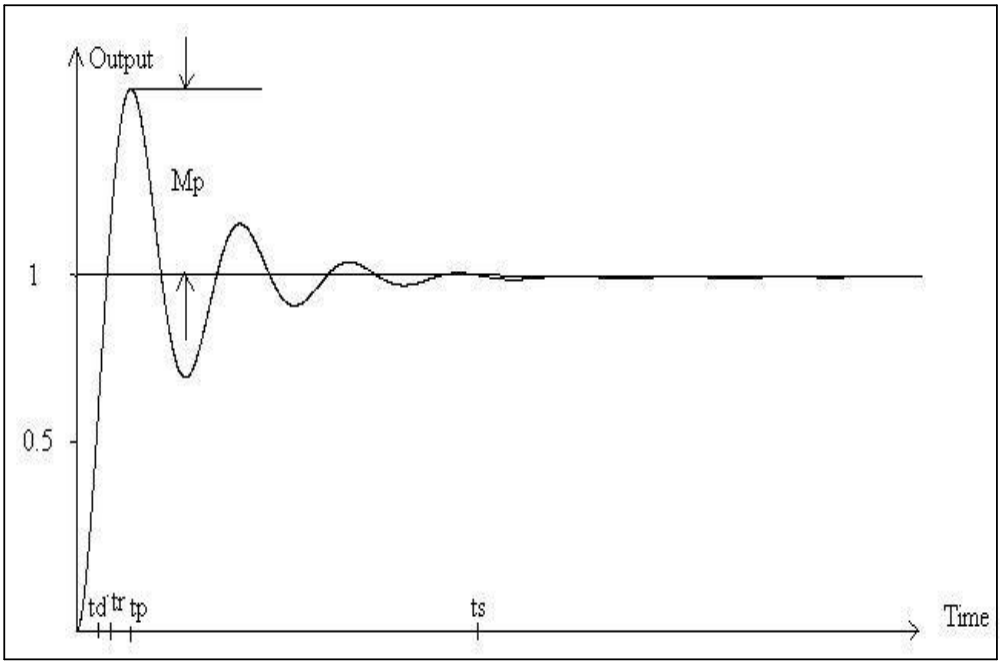

Figure 2.7. Unit step response of a second-order system

Our experiment emulator has generated the Plot in Figure 2.7. Here we can see tr, td, tp, ts and Mp in the in the unit step response.

In remote experiment system, student will determine the PID parameters of the controller according to desired transient characteristics of overall system.

The next step of the analysis is to select the appropriate method to implement the control system.

2.2.5 Two Approaches for Controllers

2.2.5.1 Microcontroller Based Control

Microcontroller is a computer that all components of computer are on a single chip. These components are control unit, arithmetic logic unit, I/O interface and memory. In microcontroller-based control, control algorithm runs on the microcontroller's memory. For the meeting the remote experimentation needs, it will be suitable to connect the microcontroller to the serial port of a PC, because most of microcontroller have serial unit for PC interfacing. This PC can be a web server or an agent for a web server.

All communication between microcontroller and this PC is obtained through RS-232 interface. Experiment plant is connected using the microcontroller I/O ports and if it is needed, the number of these ports can be increased with external chips named peripheral interface adapter (PIA).

If we want all Java solution for the remote experiment system, serial port access should be obtained using Java. Java Communications API is appropriate for this purpose. Java communications API contains support for RS232 serial ports and IEEE 1284 parallel ports. We can enumerate available ports in the system and perform asynchronous and synchronous I/O on ports (CommAPI, 2002).

Let us examine the sequence of the process. Microcontroller waits for the PID parameter at the present time. PC sends parameters and waits for the result. Result is the output of the plant. In fact, it is the output of feedback sensor. All sampled outputs are written into microcontroller memory during the experiment and transferred to the PC memory. At this time we have only raw data. Here we have two choices. One is sending the raw data to client. The other is processing

and then sending. If raw data is sent, a program at the client side should make it visualized. On the other hand, if processed data is sent, client will loose the chance of making detailed analysis. Thus both of them should be sent to the client so client will make further analysis using the raw data.

Besides, in the microcontroller market there are some sorts of microcontrollers that include Ethernet module. They do not need any external web server to send and get information, but their applications to remote experiment systems are impractical because, still we need a PC as a web server for other purposes. But it can be used as only remote instrumentation server.

As a conclusion, microcontroller based controller is one possible alternative for remote experimentation.

2.2.5.2 PC Based Control

If PC based control is used, control algorithm runs on the PC memory. Communication between PC and experiment setup is obtained with PC interface. Using a PC provides us various interface alternatives.

A general-purpose interface board is the most suitable solution if we need a lot of input and output ports. It is connected to the PCI slot of the PC. PCI (Peripheral Component Interconnect) bus standard have been developed by Intel in order to use for its own microprocessors. PCI has 32 and 64 bit data bus. Also, PCI bus supports maximum ten peripheral interfaces (Gümüşkaya, 1999). Its signaling mechanism is very complex and it requires an expertise to design a card that uses PCI bus.

ISA bus is other alternative for interfacing. This is old standard for PC interfacing. PC producers have started to leave this standard. But an old IBM PC can be used to implement ISA bus interface.

We cannot access the addresses of I/O with java. If we use PCI or ISA bus for interfacing, all java solution will be infeasible. One solution can be obtained by using a DLL that performs a communication between the hardware and java. DLLs are executable procedures that are loaded when they are needed. A DLL that performs this hardware access can be coded with a programming language

that can access hardware directly. It can be C or another language. Then Java uses this DLL with its native interface. Java native interface is used when a procedure, which is coded by using another language, is required (Schildt, 2001). But there are some drawbacks in the usage of native methods. One of them is loosing the portability because of using a native method depended on the native CPU. The other disadvantage is potential security risk. Because of this risk applets, which are small applications and one of the components of a web document that can be carried through the Internet have no right to use native methods (Schildt, 2001).

One practical solution is to use parallel port. This solution is suitable for simple applications. Parallel port has 8-bit data pins. Standard parallel port supports unidirectional data transfer. It is only used for output. Besides, other parallel port standards are PS/2, Enhanced Parallel Port (EPP), Extended Capabilities Port (ECP) permits bidirectional data transfer (Axelson, 1999). 8-bit is not enough for experiment setup interface. We should have at least two signals. Actuator signal as an output and feedback signal as an input…Thus we need to expand the I/O capacity using external chips called Peripheral Interface Adapter (PIA). Most commonly used PIA is Intel's 8085 chips. They triple the I/O capacity (Gümüşkaya, 1999). Other parallel interfaces are Small Computer System Interface (SCSI) and IEEE 488 (Axelson, 1999). IEEE 488 is important for us, because they are commonly used for laboratory equipment interfacing. Also expansion cards for IEEE488 are available.

Also serial port (RS-232) is another alternative for interfacing. But this cannot alone provide I/O operations for remote experimentation. With an external board this alternative may be utilized. But this solution is not better than the solution, which includes serially connected microcontroller as previously described. RS-485, Universal Serial Bus (USB), Fireware (IEEE 1394) and IrDA (Infrared Data Association) are some of the other serial interfaces different from the RS-232 interface (Axelson, 1999). Each has different transfer rates, maximum allowable cable lengths and maximum allowable number of device connections (Gümüşkaya, 1999).

If we want all Java solution, Java Communication API should be used. But as we state in the previous part, this API supports only RS-232 serial and IEEE

1284 parallel port standards. Thus, native methods should be used for other standards.

In this part we made a survey about controllers and interfacing alternatives. We will continue with other two components of remote experimentation.

2.3 Access Management

Doing an experiment is only one phase of the real laboratory applications. Other phases are registration, theoretical study and preliminary work including necessary simulations, laboratory quiz and report. In designing a remote system all these phases should be taken in consideration.

Access management is one of the main parts of the remote experimentation. Student's access to the system should be controlled at the each phase of the application process.

Firstly, students enter the web page of the experiment. They examine the documents about the subject of the experiment. This examination corresponds to the theoretical study before the experiment. Animations about the subject strengthen the comprehension. Everything up to here had been doing at the first ages of the distant education. Thus, animations can be considered as the first level of remote experimentation.

After theoretical study or immediately at the beginning, students should register to the system entering their name and e-mail. Session should be opened after the login process. In the session, student can view the previous documents. In addition they will have a chance of getting a quiz. This quiz corresponds to a preliminary work. If a student wants to take this quiz, a quiz page will be opened with several test questions. Results are recorded and informed to the students. Successful students will have right to download the simulation applet and to determine the experiment time suitable for them. Unsuccessful students will have no chance to continue. Until the experiment, students will have time to do simulations. In these simulations, they have a second order system to be

controlled. ω_n and ξ are the random parameters that describe the system. They should enter the PID controller parameters in order to satisfy the desired transient response characteristics such as rise time, maximum overshoot and settling time. Nowadays, making a simulation of an experiment is the most common technique at distant education of laboratory courses. Simulation process obtained by an applet can be considered as the second level of remote experimentation.

Students can have a chance to change the time of experiment if the schedule is available. Then they perform the third and the last level of remote experimentation. Doing the real experiment remotely. They send their PID parameters as they did at the simulation phase and get the drawing of the transient response function of over all system. They repeat the experiment in a given period of time. All parameters they sent are recorded to the system. According to the parameters, an evaluation of the student is made and recorded. At the end of the experiment students' accounts have expired.

2.4 Multimedia Support

Multimedia is the core of the remote experimentation process. It gives the feeling of a real experiment to the remote student. With the visual feedback, student manages the experiment and checks the experiment environment is as expected. A web camera provides this video broadcast.

Mustek 300 mini web cam has been tested using the methods suggested by Java Media Framework (JMF), which makes it possible to capture the streaming video. In order to capture and to transmit images, we have used Real Time Transfer Protocol (RTP) implemented by JMF. This protocol is widely used for videoconference systems. It is faster for the transfer of an image than the TCP/IP protocol since it uses sending of UDP datagram and does not check the good arrival of the packages because loosing packages is not significant for the

streaming video. However, JMF is not the part of Java Development Kit (JDK) and not supported by the browsers. It means that the student has to download a plug-in, which has size about 8 Mb.

The RTP protocol will be examined in order to comprehend the real-time media streams.

2.4.1 Streaming Media and Real Time Transfer Protocol

When media content is streamed to a client in real-time, the client can begin to play the stream without having to wait for the complete stream to download.

Transmitting media data across the net in real-time requires high network throughput. It is easier to compensate the lost data than to compensate large delays in receiving the data. This is very different from accessing static data such as a file, where the most important is that all of the data arrive at its destination. Thus, the protocols used for static data do not work well for the streaming media.

The HTTP and FTP protocols are based on the Transmission Control Protocol (TCP). TCP is transport layer protocol designed for reliable data communications. When a packet is lost or corrupted, it is retransmitted. The overhead of guaranteeing reliable data transfer slows the overall transmission rate. Then, another protocol should be used for streaming media.

User Datagram Protocol (UDP) is commonly used protocol for streaming media. UDP is an unreliable protocol. It does not guarantee that each packet will reach its destination. Also, there is no guarantee that the packets will arrive in the order that they were sent.

UDP is a general transport layer protocol, which application specific protocols are built on it. RTP is the Internet standard for transporting real-time data such as video and it is often used over UDP.

RTP is defined in Internet Engineering Task Force's RFC 1889. RTP enables to identify the type of data being transmitted, determine what order the packets of data should be presented in, and synchronize media streams from different sources.

An RTP session is an association among a set of applications communicating with RTP. Each media type is transmitted in a different session. For example, if both audio and video are used in a conference, one session is used to transmit the audio data and a separate session is used to transmit the video data. This enables participants to choose which media types they want to receive. So, someone who has a low-bandwidth network connection might only want to receive the audio portion of a conference.

JMF enables the playback and transmission of RTP streams through the APIs defined in the `javax.media.rtp`, `javax.media.rtp.event`, and `javax.media.rtp.rtcp` packages.

We can play incoming RTP streams locally, save them to a file or both. Also, we can implement a videoconference application that captures live audio and video and transmits it across the network using a separate RTP session. Similarly, we can record a conference for later broadcast or use a previously recorded audio in a conferencing application.

Figure shows the how RTP transmission process occurs.

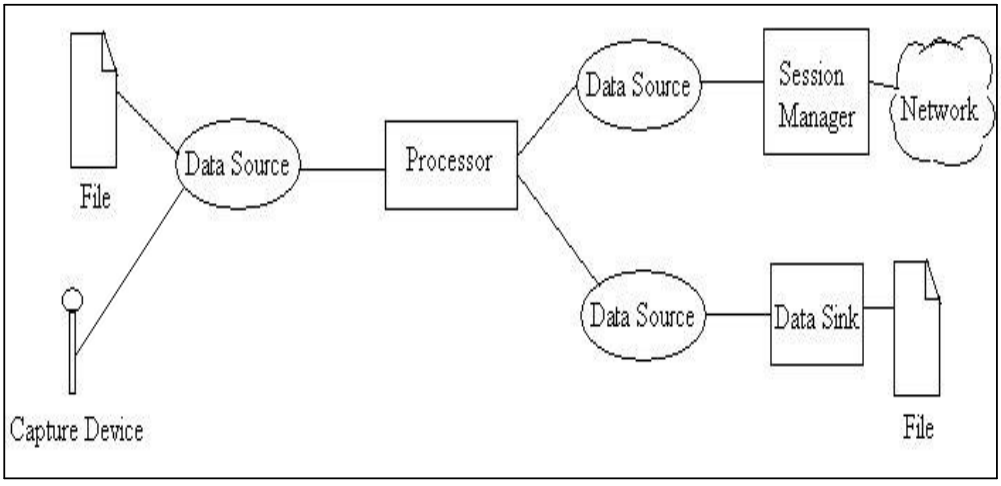

Figure 2.8. RTP transmission

In the figure, processor performs some user-defined processing on the media data, and then outputs the processed media data. JMF use data source to manage the transfer of media content. A data source encapsulates both the location of media and the protocol. Either a JMF media locator or a universal resource locator (URL) identifies a data source. Media locator is similar to a URL and can be constructed from a URL. A session manager is used to coordinate an RTP session. The session manager keeps track of streams that are being transmitted. In JMF, session manager interface defines the methods that enable an application to initialize, start participating in a session and close the entire session. A data sink is used to read media data from a data source and render the media to the destination. A particular data sink might write data to a file or write data across the network. Using JMF, data sink object are constructed through the manager class using a data source.

The simplest way to transmit RTP data is to construct an RTP data sink object using the createDataSink method of Manager class. Then we should pass in the output data source from the processor and a media locator that describes the RTP session to which the data source is to be streamed. The media locator provides the address and the port of the RTP session. To control the transmission, we can call start and stop methods on the data sink (JMFAPI, 1999).

Another way to transmit RTP data can be performed with Session Manager. Firstly, we should create a JMF Processor object and set each track format to an RTP specific format. After retrieving the output DataSource object from the processor, we should call createSendStream class on a previously created and initialized SessionManager object, passing in the DataSource object. Lastly, we should start the session manager by calling the startSession method of the SessionManager object (JMFAPI, 1999).

As a conclusion Java Media Framework makes easier to capture and transmit the video. Using JMF can be the part of the all-Java solution in remote experimentation.

Chapter 3

REMOTE EXPRIMENT SYSTEM PROPOSED FOR IYTE

According to the basic requirements that we have mentioned, such architecture can be proposed for remote experiment system. We must have an interfacing computer to interface the experiment system. Database server is proposed to use in access management and experiment scheduling activities. Multimedia server is an interface for multimedia components such as web cam. All these servers can be executed either on a single computer or on different computers. Architecture should be checked if it is feasible or not for IYTE.

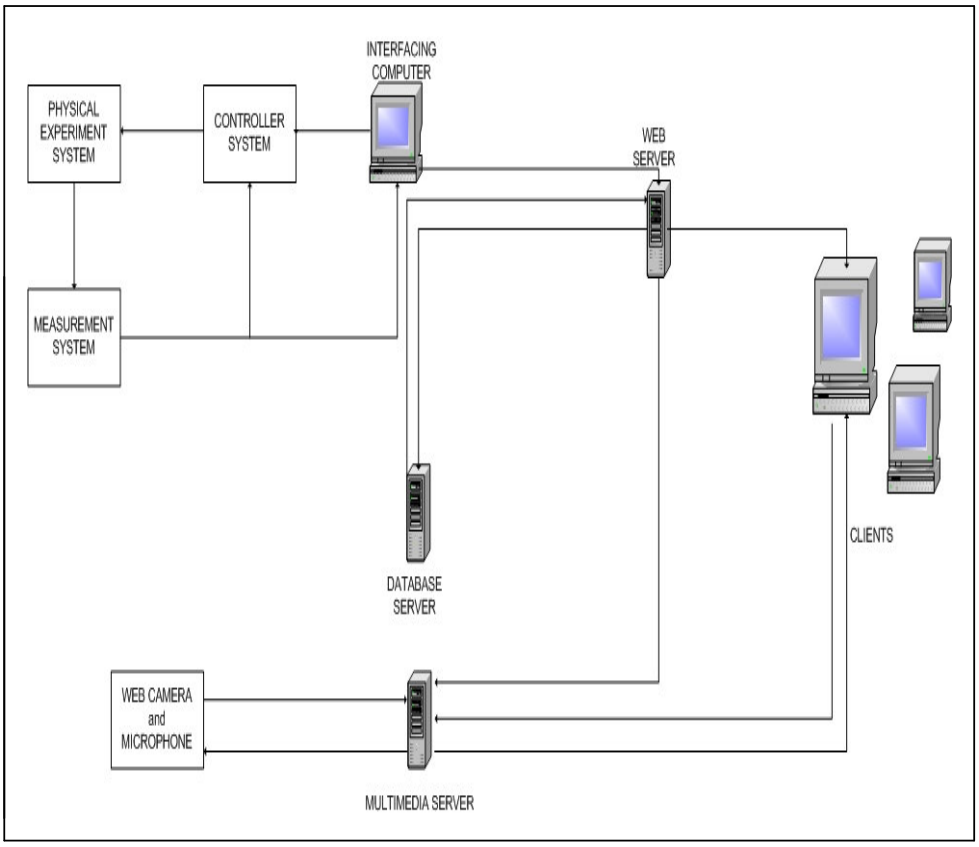

Figure 3.1. Proposed Remote Experiment Architecture

Up to here, the three basic components of a remote experiment system have been examined. Access management and multimedia components can be applied to any laboratory course. The critical component is design of the controller. Therefore in order to determine the environment that the control algorithm will execute on, we have done an inventory research.

3.1 Feasibility Study

In IYTE, Computer engineering department there is a computer architecture laboratory. In this laboratory Intel's 8086-microprocessor development kits, 8051 microcontrollers, digital oscilloscopes and lots of EPROM and Peripheral Interface Adapter (PIA) chips are the main laboratory equipments.

8086 development kit has a ROM-based monitor program for simple interaction (MTS86-C). Using its RS-232 serial interface, data transmission can be done between kit and PC. It has Analog to Digital and Digital to Analog Converter. Also its three PIA is suitable for experiment interfacing. In the case of remote experiment, control parameters can be taken and the output can be sent using the serial port. All communication is performed through this port.

To use 8051 microcontroller is another alternative solution. It has ROM based BASIC interpreter for programming (Axelson, 1997). This interpreter gets the code from the PC through the serial port. It is unnecessary for our needs. Instead, it will be better to use an EPROM for programming needs.

Up to here, everything seems agreeable, but we left all control to the microprocessors. If we want to design a controller other than a PID controller, all software should be changed. We loose the flexibility and we restrict ourselves with a single controller algorithm.

If we examine the PC based controller solution, available PC Interfaces should be searched. At the Computer Engineering Department we don't have any interface board for PCI or ISA slot. Then we have decided to design an emulator on PC, that emulates the experiment hardware and to concentrate on the protocol between this emulator and student. So when we have chance to get any PCI card in the future, new hardware can be substituted with the emulator and the same

protocol can be used. This emulator idea can be seen as the first step of building the remote experiment system.

3.2 Emulator Solution

Emulators imitate real systems. In this emulator, we are trying to emulate the experiment plant and PID controller. For the experiment system, we will use the mathematical model, which has been explained in the controller design chapter previously.

That was the second-order system having two parameters that describe its characteristics. Damping ratio ξ and undamped natural frequency ω_n. The emulator will produce these parameters randomly. Then emulator sends the plot of unit step response of the system to the student for system identification. Emulator waits for the PID parameters. After getting the PID parameters, emulator executes the controller process under a unit step input. Then emulator sends the response curve as a JPEG image. JPEGImageEncoder class, which belongs to com.sun.image.codec.jpeg package of Java, can be used for this purpose.

Fourth order Runge-Kutta method (Karagöz, 2001) has been used for the execution of the overall controller system that can be defined with third order differential equation.

Then, after the inclusion of the emulator, our architecture has changed.

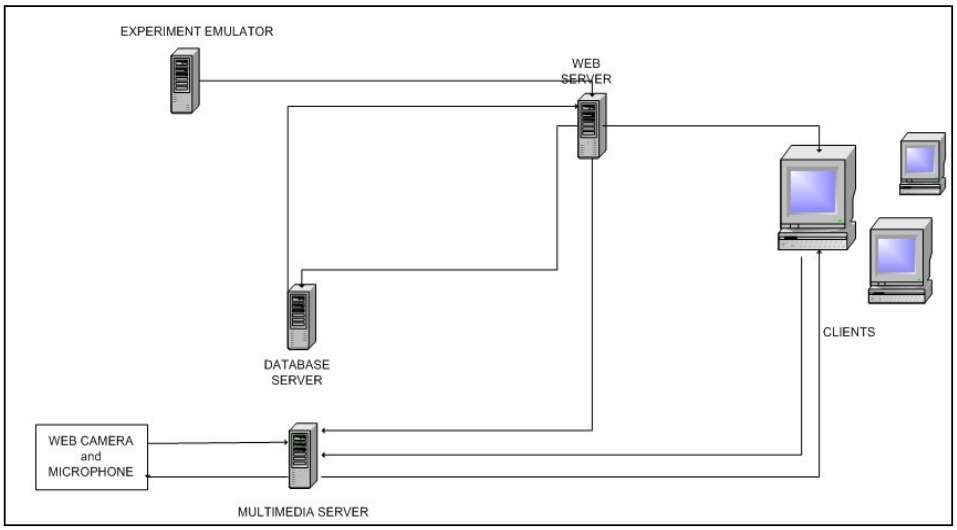

Figure 3.2. Remote Experiment System with the Emulator

3.3 Mathematical Model of Experiment Emulator

In emulator, parallel RLC circuit was used as an experiment plant. This circuit includes resistor, capacitor and inductor in parallel.

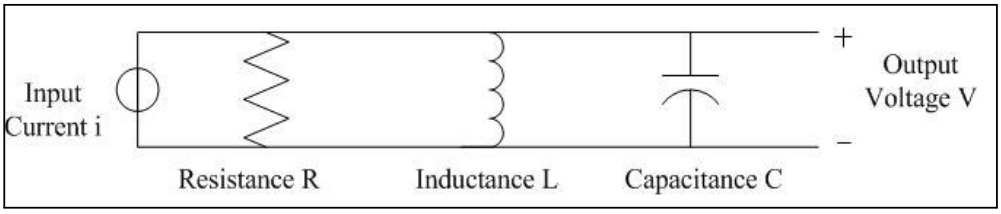

Figure 3.3. RLC circuit

After adding PID controller, whole emulator system's block diagram can be seen in figure 3.4.

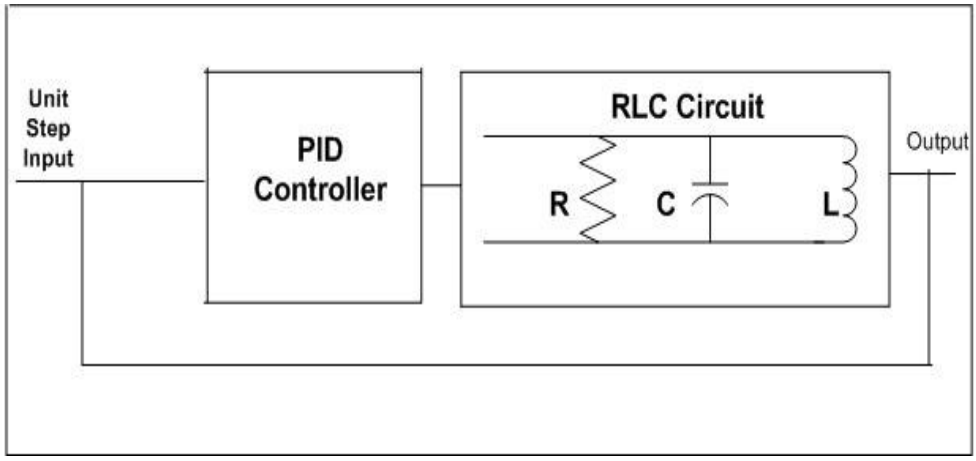

Figure 3.4. Emulator system block diagram

In order to describe emulator system with its tranfer function, differential equations of PID controller and RLC circuit should be given.

$$u(t) = K_p * e(t) + (K_p/T_I) * \int_0^t e(t)\,dt + K_p T_d * (de(t)/dt) \qquad \text{(eq. 3.1)}$$

In equation 3.1, e(t) is error input, K is proportional parameter, Ti is integral parameter and Td is derivative parameter.
Acoording to figure 3.3;

$$I=V/R + (1/L) \int_0^t V\,dt + C\,(dv/dt) \qquad \text{(eq. 3.2)}$$

$$(d^2V / dt^2) + (1/RC)(dV/dt) + (1/LC)V=i \qquad \text{(eq. 3.3)}$$

When we take Laplace tranforms of the differential equations,

For RLC circuit:

$$V(s)[s^2+(1/RC)s+(1/LC)]=I(s) \qquad \text{(eq. 3.4)}$$

For PID controller

$$U(s)= K_p * [1 + (1 / (T_i * s)) + (T_d * s)]E(s) \qquad \text{(eq. 3.5)}$$

Then transfer functions of block diagram in figure 3.4,

For RLC circuit:

$$T1(s)=V(s) / I(s) = RLC / [RLs^2 + Ls + R] \qquad \text{(eq. 3.6)}$$

For PID controller

$$T2(s) = U(s) / E(s) = K_p * [1 + (1 / (T_i * s)) + (T_d * s)] \qquad \text{(eq. 3.7)}$$

Open loop transfer function of the system To(s) is,

$$To(s) = T1(s) * T2(s) \qquad \text{(eq. 3.8)}$$

$$To(s) = (KpTdTiRLC\ s^2 + KpTiRLC\ s + KpRLC) \,/\, (RLCTi\ s^3 + TiL\ s^2 + RTi\ s)$$

(eq. 3.9)

When feedback is added, closed loop transfer function Tc(s) is,

$$Tc(s) = To / (1+To)$$ (eq. 3.10)

$$Tc(s) = \frac{Y(s)}{U(s)} = \frac{(KpTdTiRLC\ s^2 + KpTiRLC\ s + KpRLC)}{RLCTi\ s^3 + (RLCKpTdTi+TiL)\ s^2 + (RLCKpTi+RTi)\ s + KpRLC}$$

(eq. 3.11)

In order to solve this transfer funtion with computer, state space definition of the system should be obtained. Thus, 3 first order differential equation can be solved simultaneously with Runga-Kutta method.

The state space representation for the transfer function

$$T(s) = \frac{Y(s)}{U(s)} = \frac{b_0 s^n + b_1 s^{n-1} + \ldots\ldots\ldots\ldots\ldots\ldots + b_{n-1} s + b_n}{s^n + a_1 s^{n-1} + \ldots\ldots\ldots\ldots\ldots\ldots + a_{n-1} s + a_n}$$

(eq. 3.12)

can be given by equations (eq. 3.13) and (eq. 3.14)

•

$$\dot{X} = \mathbf{A}x + \mathbf{B}u \qquad \text{(eq. 3.13)}$$

$$Y = \mathbf{C}x + \mathbf{D}u \qquad \text{(eq. 3.14)}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \\ \\ x_{n-1} \\ x_n \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & .. & 0 \\ 0 & 0 & 1 & .. & 0 \\ .. & .. & .. & .. & \\ & & & & \\ & & & & \\ 0 & 0 & .. & & 1 \\ -a_n & -a_{n-1} & .. & & -a_1 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \\ \\ \\ \\ \beta_{n-1} \\ \beta_n \end{bmatrix}, \quad \mathbf{C} = [\,1 \quad 0 \,..\, \ldots \quad 0\,], \quad \mathbf{D} = [\,\beta_0\,]$$

Our transfer function is third order then its state space form is,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_3 & -a_2 & -a_1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} .u$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \beta_0 .u$$

a1, a2, a3 can be obtained from the equations.

β_0 , β_1 , β_2 and β_3 can obtained from the following equations,

$\beta_0 = b0$ (eq. 3.15)

$\beta_1 = b1\text{-}a1\ \beta_0$ (eq. 3.16)

$\beta_2 = b2\ \text{-}a1\ \beta_1 - a2\ \beta_0$ (eq. 3.17)

$\beta_3 = b3 - a1\,\beta_2 - a2\,\beta_1 - a3\,\beta_0$ (eq. 3.18)

After getting state space form of our emulator system, Runga-Kutta method can be applied to these 3 derivative equations.

•

$x_1 = x_2 + \beta_1 u$ (eq. 3.19)

•

$x_2 = x_3 + \beta_2 u$ (eq. 3.20)

•

$x_3 = x_2 = -a_3 . x_1 - a_2 . x_2 - a_1 . x_3 + \beta_3 u$ (eq. 3.21)

•

$x_1 = \frac{dx}{dt} = f(t, x)$ Assume that in t-x plane (t_i, x_i) is known and we wish to find point (t_{i+1}, x_{i+1}). The incremental time $t_{i+1} - t_i = h$ is the time interval for computation, or the sampling period.

k1, k2, k3 and k4 are Runga-Kutta parameters which are the change of the x value at $t = t_{i+1}$.

$k1 = h*f(t_i, x_i)$ (eq. 3.22)

$k2 = h*f(t_i + \frac{1}{2} h, x_i + \frac{1}{2} k1)$ (eq. 3.23)

$k3 = h* f(t_i + \frac{1}{2} h, x_i + \frac{1}{2} k2)$ (eq. 3.24)

$k4 = h* f(t_i + h, x_i + k3)$ (eq. 3.25)

When we obtain the weighted average of k1, k2, k3 and k4, where the weights are 1, 2, 2 and 1 respectively,

$$\Delta x_i = x_{i+1} - x_i = (1/6) * (k1 + 2k2 + 2k3 + k4) \qquad \text{(eq. 3.26)}$$

Then the value of x_{i+1} can be given by

$$x_{i+1} = x_i + \Delta x_i = x_i + (1/6) * (k1 + 2k2 + 2k3 + k4) \qquad \text{(eq. 3.27)}$$

This equation is called the fourth order Runga-Kutta equation, because it involves four values of k's.

3.4 Implementation of Experiment Emulator

Runga-Kutta equation is used for the simultaneous computation of x_1, x_2 and x_3 with the help of eq. 3.19, 3.20 and 3.21. Program is coded in java and java server page is used. Tomcat is used as jsp server. Apache is used as web server. Servers executed on a PC having Windows XP operating system and 128 MB memory.

In program, three methods were used for defining the three states of system and one method for Runga-Kutta method. After server is started and URL is entered, such form is shown as in figure 3.5.

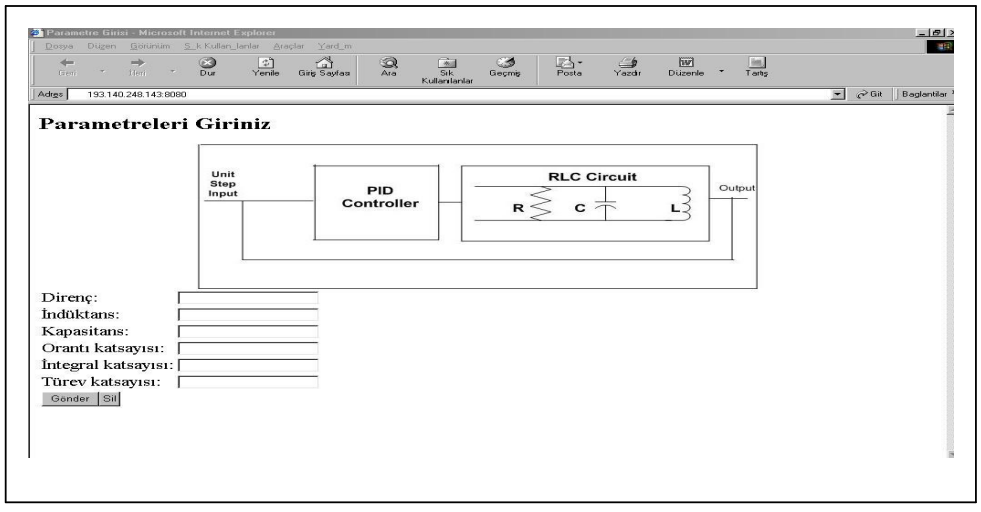

Figure 3.5. Html form for Emulator parameters

After entering the parameters, v-t plot is drawn, where v is output voltage and t is time.

Sample figure is shown in figure 3.6.

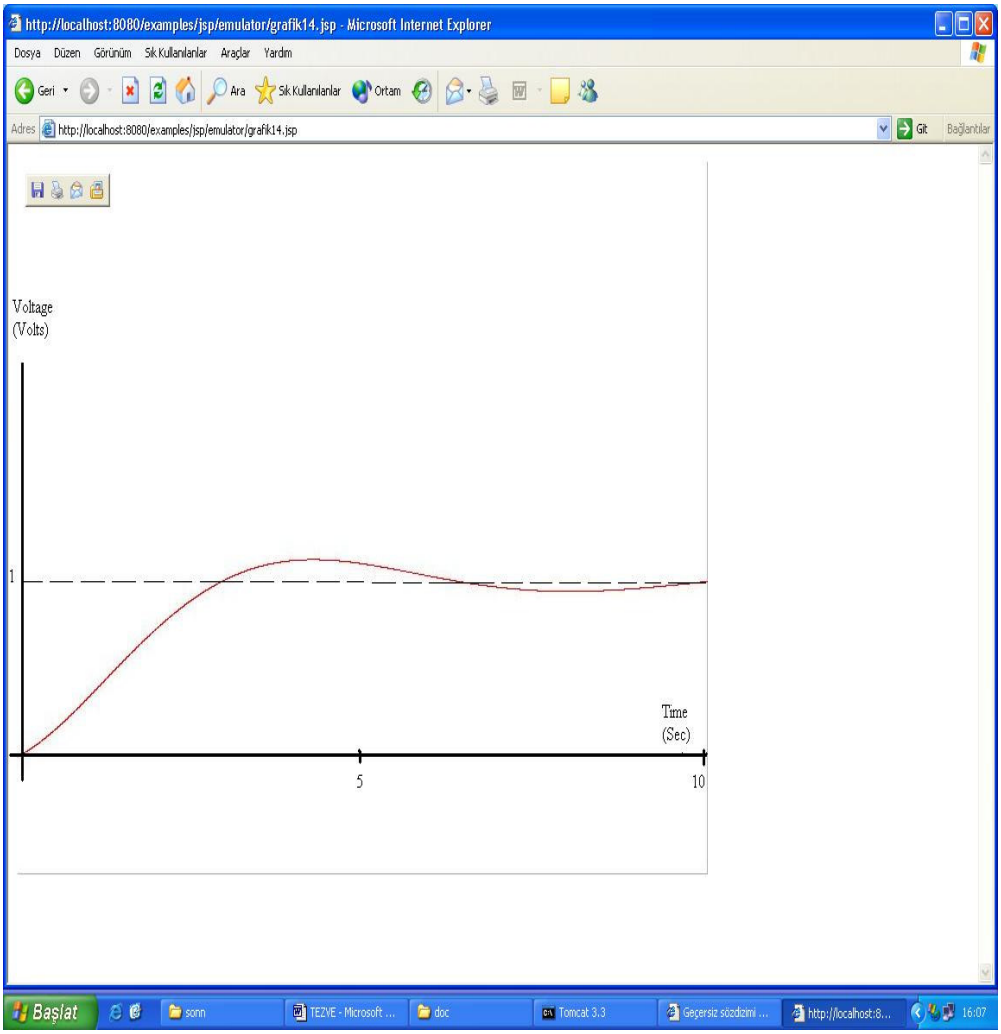

Figure 3.6 V-t plot of emulator

At that computation incremental time, h, is taken as 0.001. Plot is drawn for 10 seconds of voltage signal, where R=0.4 Ohm, C=1 Farad, L=1 Henry, P=1, Ti=5 and Td=0.2.

Chapter 4

OBJECT-ORIENTED ANALYSIS and DESIGN with UML

4.1 UML

Object-Oriented software development techniques have gone through three stages of evolution.

1. Object-Oriented programming languages were developed and began to use.
2. Object-Oriented analysis and design techniques were produced to help in the modeling business, the analysis of requirements and the design of software systems.
3. Unified Modeling Language (UML) was designed to bring together the best features of a number of analysis and design techniques and notations to produce an industry standard (Bennett, Skelto & Lunn, 2001).

UML is a visual language that provides a way for people who analyze and design object oriented systems. UML includes a lot of components for different purposes. Two of them are Use Case and Collaboration. These will help us during the analysis of the system.

4.1.1 Use Case

Use Cases are descriptions of the functionality of the system from the users' perspective. Use Case diagrams show three aspects of the system: actors, Use Cases and the system boundary. Actors represent the roles that people or devices take on when communicating with the particular Use Cases in the system (Bennett, McRobb & Farmer, 2002). We have two actors in our system. Student and instructor…

4.1.2 Collaboration

In Object-Oriented systems, the objects working together produce the functionality. In order to work together, objects need to communicate with one another and they do this by passing messages. This process is called as Collaboration. In the UML specification, Collaboration is described as something that defines a set of participants and relationships that are meaningful for a given set of purposes (Bennett, McRobb & Farmer, 2002).

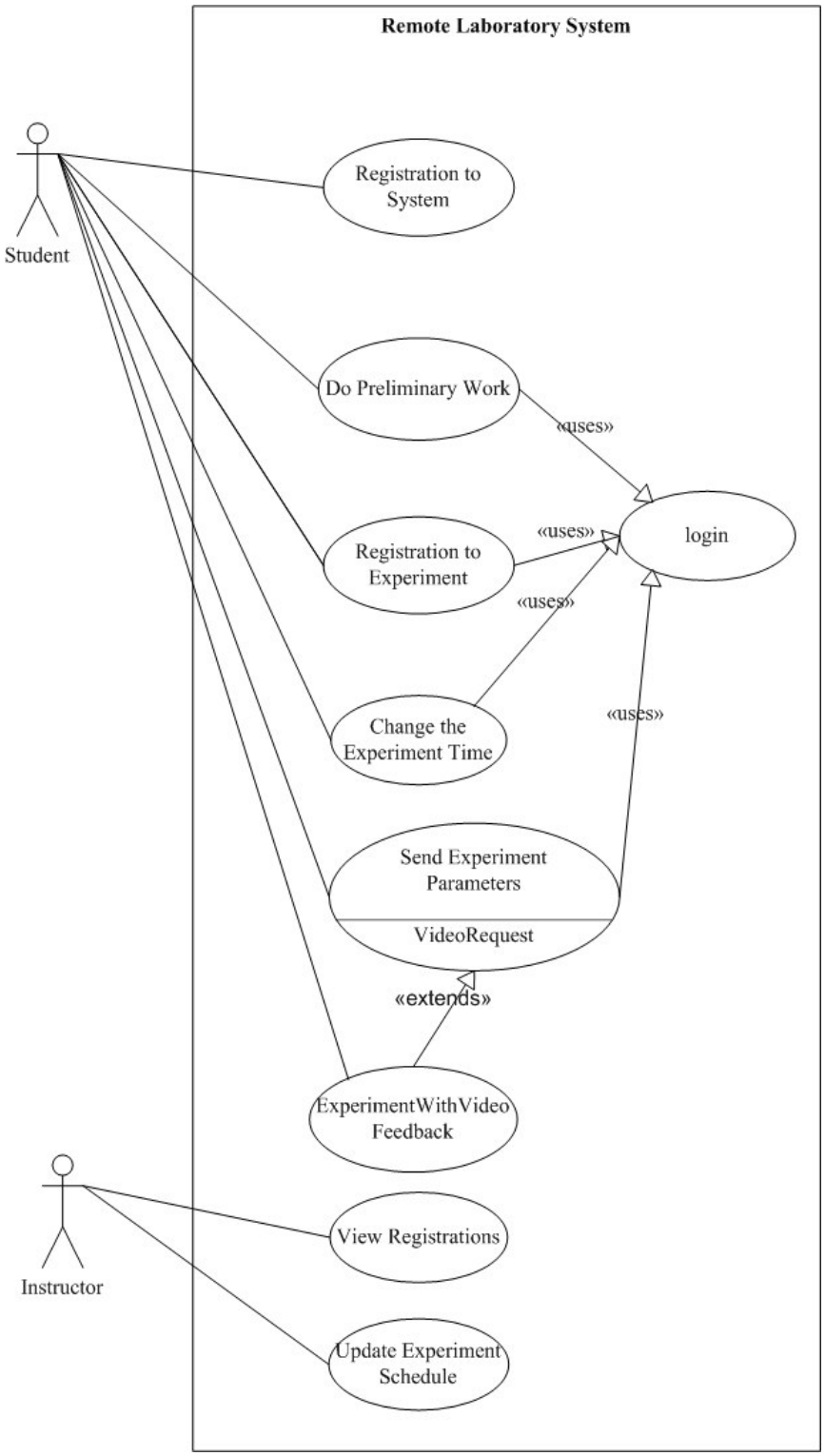

Figure 4.1. Use Case Diagram of The System

4.2 Use Case and Collaboration Diagrams of the Remote Experiment System

Preceeding figure shows the set of all Use Cases for the system. Student actor has seven Use Case. Five of them use another Use Case and one of them is extended using another one. Instructor actor has two Use Case. Each Use Case will be explained thorough their Collaboration diagrams. After we combine all information comes from collaborations, Class Diagram of the system will appear.

4.2.1 Registration to the System Use Case

When Student wants to make his or her registration through a Welcome page of the System, a registration form is opened. Student enters name, e-mail address, login name, password and other necessary information. After student approves this information, a registration control mechanism validates the registration information, performs registration and prompts user that registration has been done.

4.2.2 Collaboration Diagram for the Registration to the System Use Case

Three boundary classes, two control classes and one entity class are proposed for the collaboration of the Registration to the System Use Case.

Boundary classes are WelcomePageUI, RegistrationFormUI, and ApprovalWindow.

WelcomePageUI is responsible to generate a WelcomePage user interface. It should contain a link for the static page, in which all theoretical explanations have been presented. Also it should contain separate links for login and system registration. When the login link is activated, a login form is opened containing login name and password text fields. And when the system registration link is activated, Registration form is opened as in our use case.

RegistrationFormUI is responsible to perform the activities related to the registration form. It has form fields and approve link. Student fills the form and approves them using the approve link.

After the registration process has been established, ApprovalWindow is opened in order to prompt user that registration has been done.

Control classes are WelcomePageControl and RegistrationControl classes.

Here, WelcomePageControl gets the registration request from the WelcomePageUI and starts the RegistrationFormUI.

RegistrationControl gets the student information. After validating, it makes the registration sending them to the Student entity.

We have one entity class. Student entity has all attributes about students.

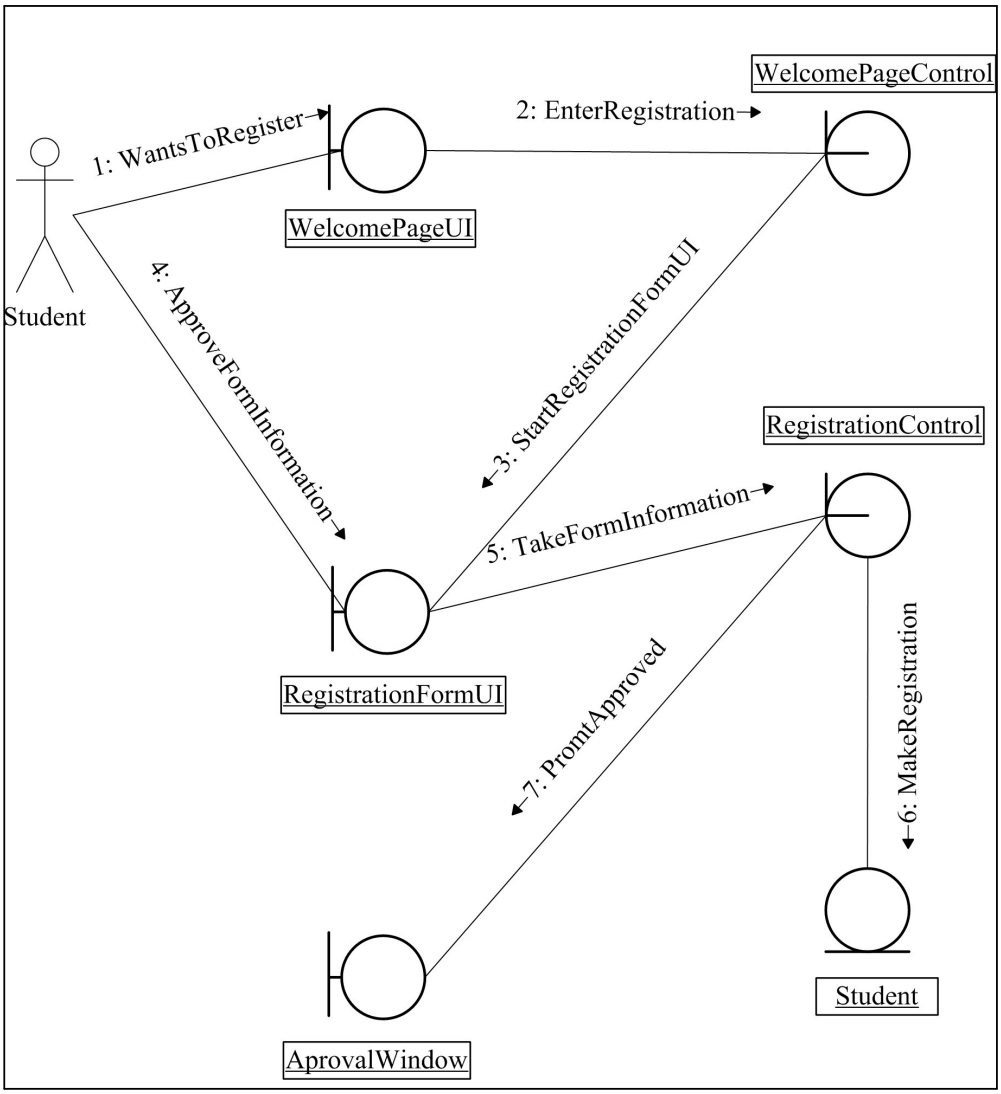

Figure 4.2. Collaboration Diagram for Registration to the System Use Case

The Sequence of operations:

1.WantsToRegister: Students activates the SystemRegistration link.

2.EnterRegistration: WelcomePageUI object calls EnterRegistration method of WelcomePageControl object.

3.WelcomePageControl object calls StartRegistrationFormUI method of RegistrationFormUI object.

4.Student fills the registration form and approves the information.

5.RegistrationFormUI object calls TakeFormInformation method of RegistrationControl object.

6.RegistrationControl object checks whether student has entered valid information and then makes registration by calling MakeRegistration method of Student object.

7.RegistrationControl object prompts user by calling PromptApproved method of ApprovalWindow object.

4.2.3 Login Use Case

When student activates the login link at the WelcomePage, Login form is generated. Student enters login name and password and approves them activating the approve link of the Login form. Login name and password are controlled and a session is started. Afterwards, one of the possible three interfaces is opened according to the state of the student. If student did not do the preliminary work, UserInterfaceA is opened. If student did the preliminary work but did not make the experiment registration, UserInterfaceB is opened. If student did the preliminarywork and made the experiment registration, UserInterfaceC is opened.

4.2.4 Collaboration Diagram for the Login Use Case

Five boundary classes, three control classes and three entity classes are proposed for the collaboration.

Boundary classes are WelcomePageUI, LoginUI, UserInterfaceA, UserInterfaceB and UserInterfaceC.

In this collaboration Login link of the WelcomePage is activated.

LoginUI generates the login form and send the information to the LoginControl object.

UserInterfaceA consists of a link for Preliminary Work. It is constructed, if the student did not do the Preliminary Work.

UserInterfaceB shows the Preliminary Work grade and it has a link for experiment registration. It is constructed, if the student did not make the experiment registration after doing the Preliminary Work.

UserInterFaceC shows the Preliminary Work grade and the experiment time. Also, it has a links for changing the experiment time and doing the experiment. It is constructed if the student made the experiment registration. If the students login at the experiment duration, this interface is generated again. Students cannot login after the end of the experiment time.

Control classes are WelcomePageControl, LoginControl and StateControl.

WelcomePageControl gets the login request from the WelcomePageUI and starts the LoginUI.

LoginControl gets the login name and password from the LoginUI and checks if they are valid by calling GetLoginApproval method of the Student object. This method returns the student id of the student. Then LoginControl starts a session for this student by calling the StartSession method of the Session object.

The Session object constructs StateControl object. Calling the GetState method of the student object controls the state of the student. If student did not do the Preliminary Work, StateControl starts the UserInterfaceA. If the student did the Preliminary but did not do the experiment registration, StateControl starts the UserInterfaceB by getting the Preliminary grade from the Student object. If the student did the experiment registration, UserInterfaceC is started getting both preliminary grade and registered experiment time.

Session object is active during the all session. In this collaboration it is used for registering the starting time of the session to the Student object's LastSessionTime variable.

The need for a session revealed a session concept automatically during the analysis. Java Server Page can be used for the session needs (Pekgöz, 2002).

We can extract from this collaboration that the Student object must include the Preliminary grade and the experiment time as an attribute.

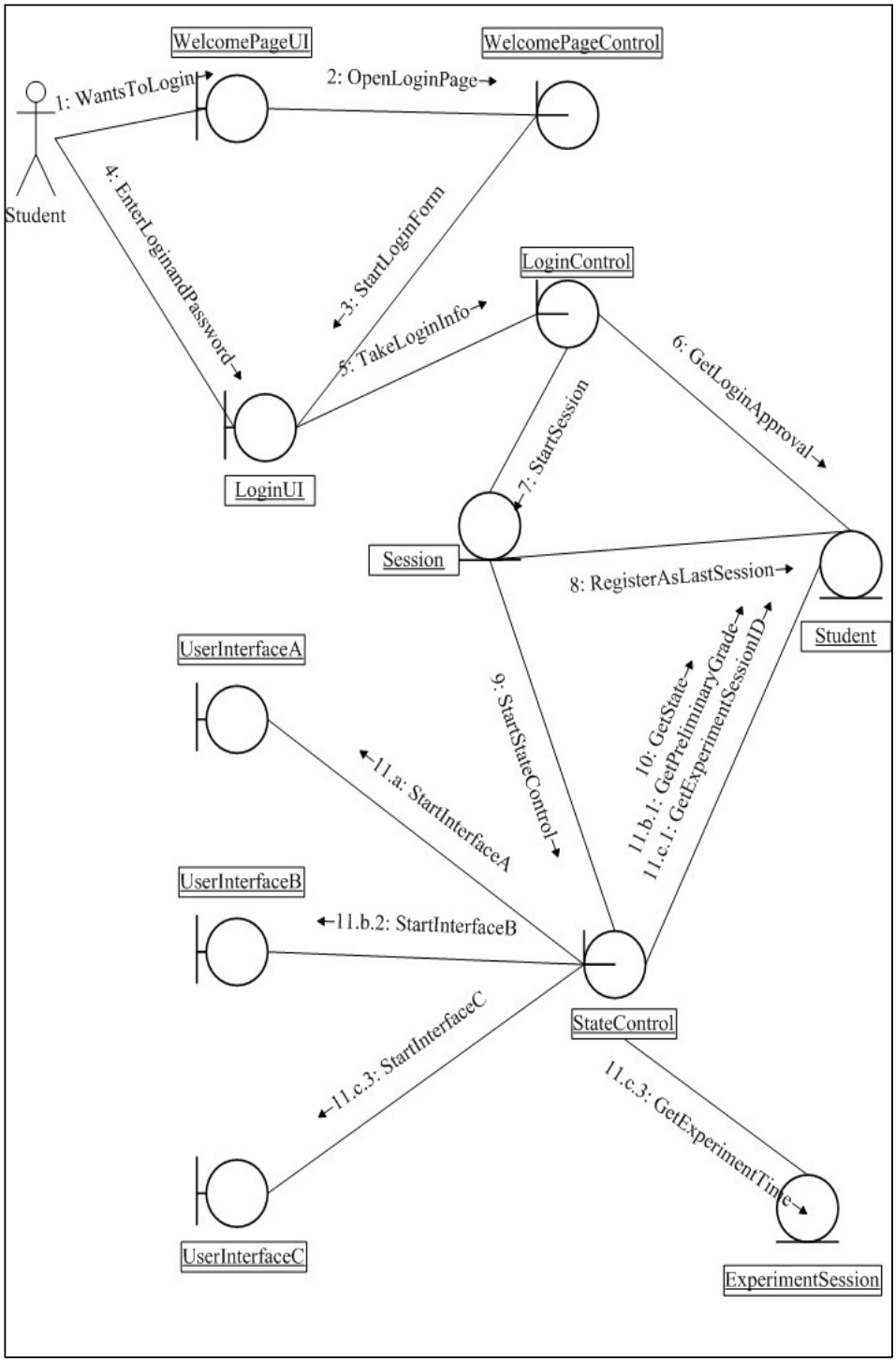

Figure 4.3. Collaboration Diagram for Login Use Case

4.2.5 Do Preliminary Work Use Case

Students enter the Preliminary Work by activating the PreliminaryWork link of the UserInterfaceA. They get Preliminary test and approve their answers. Control mechanism checks the answers and register the grade to the student entity. If the grade is enough to continue, this control mechanism starts the UserInterfaceB for experiment registration. Otherwise grade is displayed and session is closed.

4.2.6 Collaboration Diagram for the Do Preliminary Use Case

We have four boundary classes, one control and three entity classes.

Boundary classes are UserInterfaceA, PreliminaryQuizForm, UserInterfaceB and LowGradeUI.

UserInterfaceA includes the PreliminaryWork link. UserInterfaceB includes the quiz grade and experiment registration link. PreliminaryQuizForm has test questions, necessary fields for choices and approval link. LowGradeUI has only the quiz grade.

Contol class is the PreliminaryControl. It takes the request from the UserInterfaceA and starts the PreliminaryQuizForm interface taking questions from the PreliminaryQuiz entity. PreliminaryQuizForm sends the answers. Control class checks the answers and according to the grade activates the UserInterfaceB or LowGradeUI.

PreliminaryQuiz entity has quiz questions and answers. Other two entity classes are the session class and the student class.

Sequence of the operations:

1.wantsToDoPreliminaryWork: Students activates the PreliminaryWork link of the UserInterfaceA.

2.openPreliminary: UserInterfaceA sends this request to the PreliminaryControl.

3.getQuestions: PreliminaryControl gets questions.

4.showQuizForm: PreliminaryControl starts the Quiz interface.

5.sendAnswers: Student fills the quiz form and approves the form.

6.getAnswers: Quiz interface sends the answers.

7.getStudentID: PreliminaryControl gets the student id from the Session entity.

8.calculateGrade: calculateGrade method of the PreliminaryQuiz calculates the grade Student's answers are the input parameters of the calculateGrade method and method returns the grade checking with the correct answers.

9.1. setPreliminaryGrade: PreliminaryQuiz objects registers the grade to the Student entity.

9.2.a.startInterfaceB: PreliminaryControl starts the UserInterfaceB, if the grade is enough to continue. Then one of the scenarios ends.

9.2.b.1 closeSession: PreliminaryControl closes the session, if the grade is not enough.

9.2.b.2 expireAccount: Control object expires the account calling expireAccount of the Student object.

9.2.b.3 startLowGradeUI: Control object starts the LowGradeUI and the other scenario end.

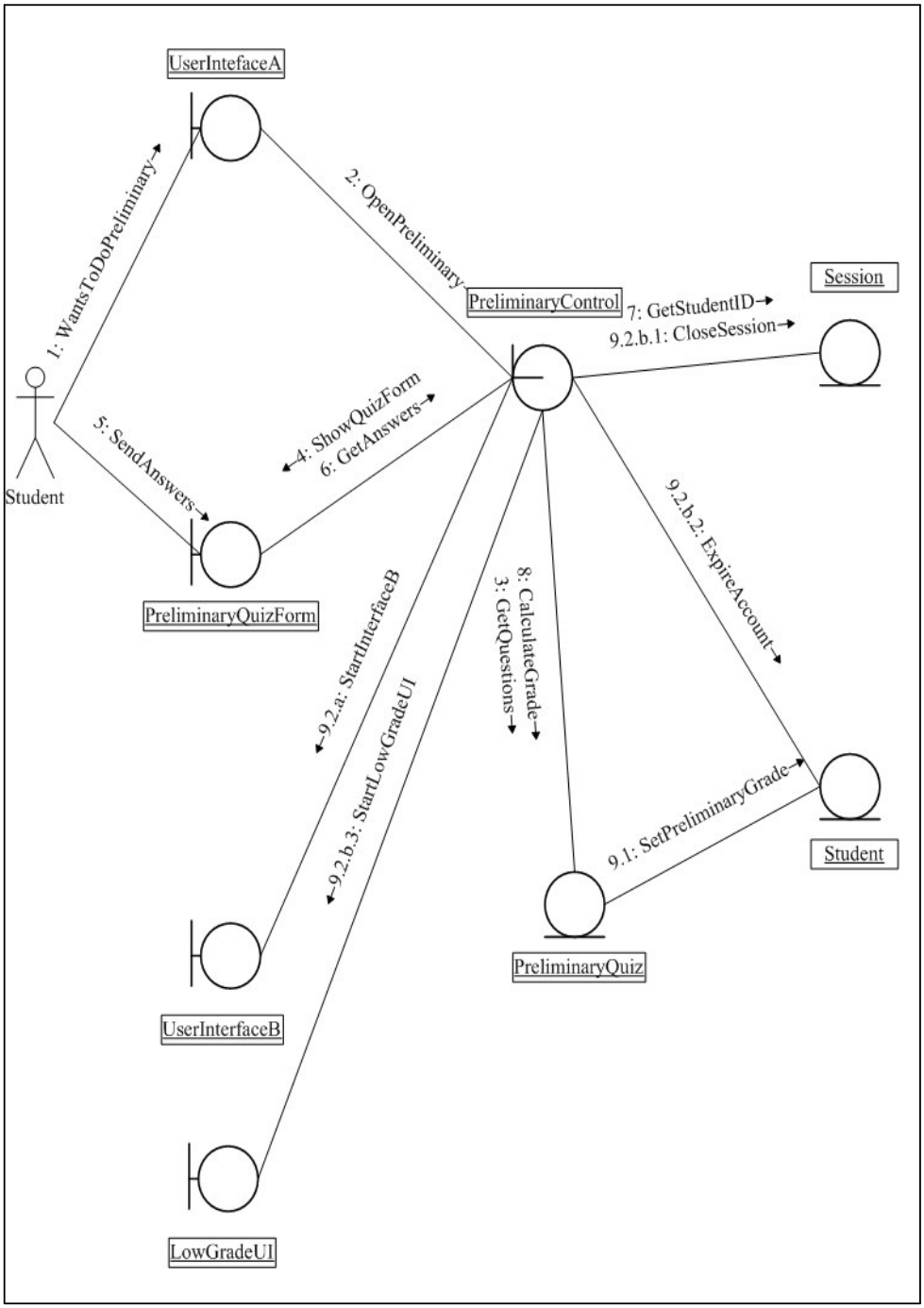

Figure 4.4. Collaboration Diagram for Do Preliminary Use Case

4.2.7 Registration to Experiment Use Case

Student starts the registration process through the UserInterfaceB activating the Experiment Registration link. Available experiment sessions are shown to the student. Student selects the appropriate session.

4.2.8 Collaboration Diagram for the Registration to Experiment Use Case

We have three Boundary classes, three entity classes and one class in the collaboration.

Boundary classes are UserInterfaceB, Schedule and UserInterfaceC.

Schedule shows the timetable of available experiment sessions. It has approval link.

Our control class is RegistrationControl. We used the same class in the collaboration of RegistrationToSystem Use Case.

Entity classes are Session, Student and ExperimentSession.

Sequence of the process is:

1.wantsToRegisterToExperiment: Student activates the ExperimentRegistration link of the UserInterfaceB.

2.wantsToregister: UserInterfaceB calls the wantsToRegister method of RegistrationControl object.

3.getAvailableTime: RegistrationControl gets the available experiment sessions from the ExperimentSession entity.

4.showAvailableTime: RegistrationControl object calls this method in order to activate the Schedule Interface.

5.enterAvailableTime: Student enters the appropriate experiment time and activates the Approval link.

6.sendTime: Schedule interface sends time to the RegistrationControl.

7.getStudentID: StudentId is taken from the Session entity.

8.recordStudentID: StudentID and selected time is sent to the ExperimentSession.

9.recordExpSessionID: After ExperimentSession object constructs the Student object with given StudentID, ExpSessionID is recorded to the student entity.

.

10.getPreliminaryGrade: RegistrationControl gets the preliminary grade from the Student entity.

11.startInterfaceC: RegistrationControl object starts the interfaceC.

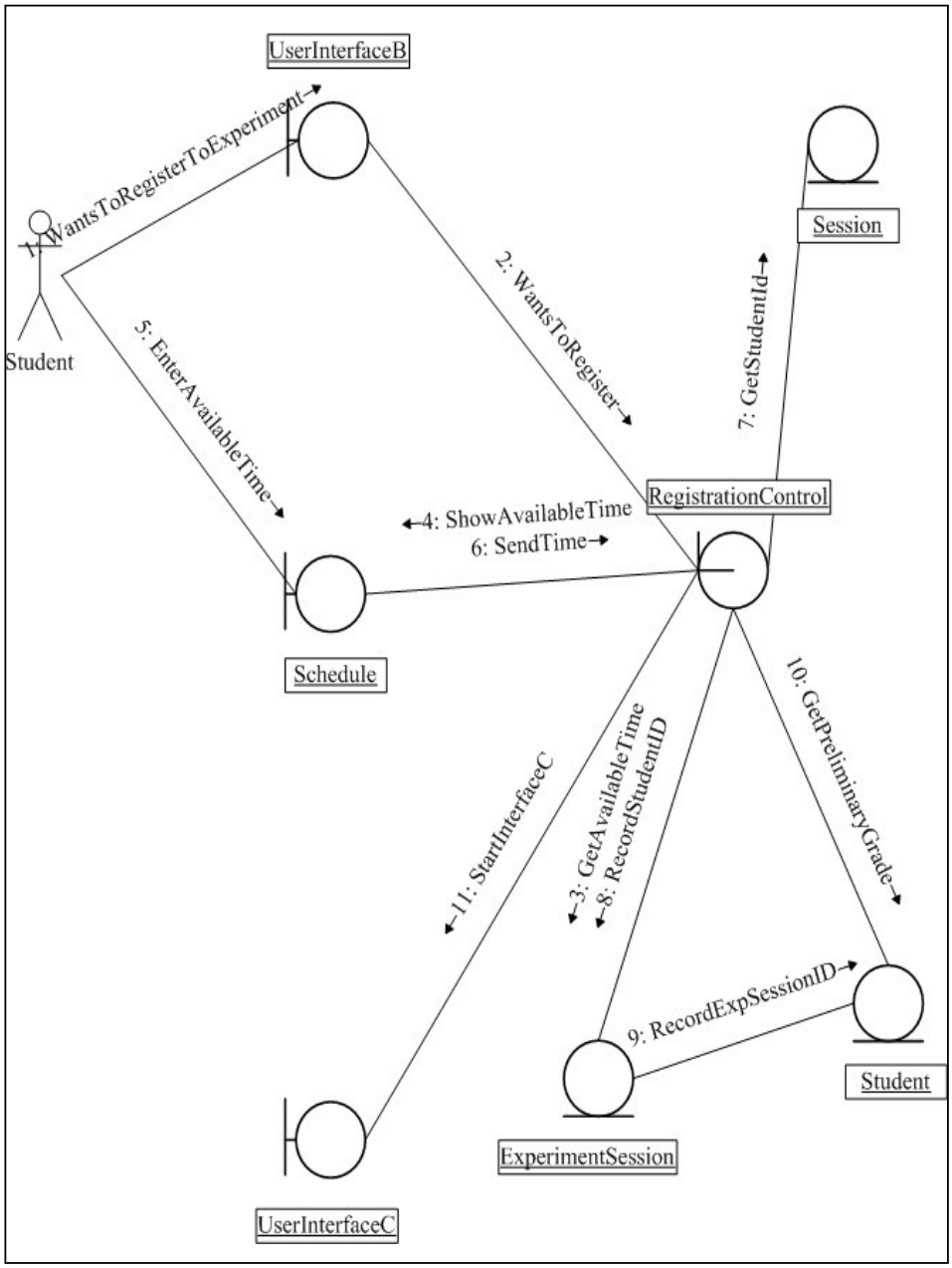

Figure 4.5. Collaboration Diagram for Registration to Experiment Use Case

4.2.9 Change the Time of the Experiment Use Case

Students can change their experiment time before the experiment through the UserInterfaceC.

4.2.10 Collaboration Diagram for the Change the Time of the Experiment

It is similar to the Collaboration Diagram of the Experiment Register Use Case. The only difference is starting interface is UserInterfaceC.

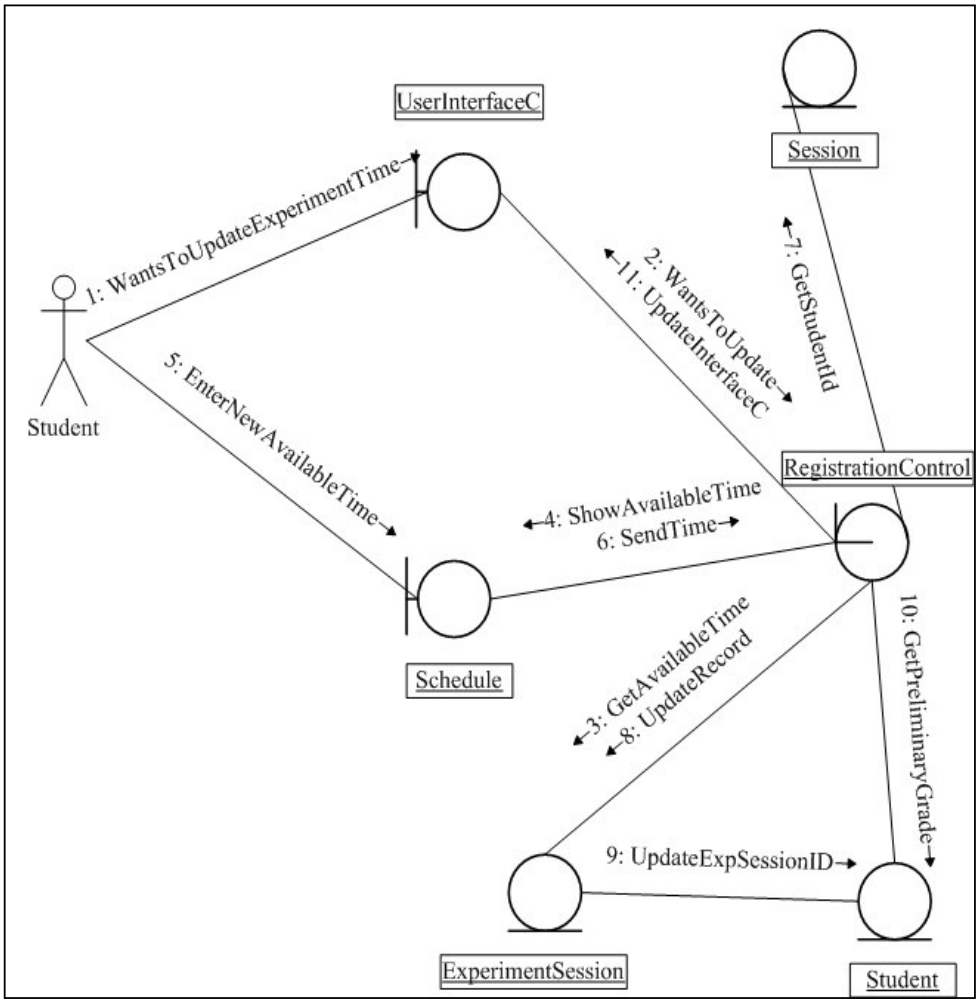

Figure 4.6. Collaboration Diagram for the Change the Time of the Experiment Use Case

4.2.11 Send Experiment Parameters Use Case

Experiment Emulator produces two system parameters ξ and ω_n randomly. The probability distribution of these parameters is uniform and values change between reasonable limits. The plot of unit step response of the system is sent to the student with transient system characteristics such as maximum overshoot, rising and settling times. In addition student gets the desired values of transient system characteristics. Student does the system identification making necessary calculations and estimates the system parameters. According to desired unit step response, student determines the PID parameters of the controller. Simulation program can be used in order to check correct PID parameters. After sending PID parameters, the Experiment Emulator generates a plot of unit step response.

System and controller parameters are recorded to the Student entity. If the student's transient system characteristics are close to the desired characteristics, experiment grade of the student will be high.

Also, a visual feedback is supplied to the student. For the emulator case, it does not matter. But when the emulator is changed with the real experiment system, similar collaborations can be used.

4.2.12 Collaboration Diagram for the Send Experiment Parameters Use Case

We used four Boundary classes, four entity classes and two control classes.

Boundary classes are UserInterfaceC, ExperimentInterface, ResponsePlot and the VideoInterface.

ExperimentInterface has text fields for PID parameters and the plot of unit step response of the second-order system. Also, desired transient response characteristics are given through this interface. Extended ExperimentWithVideoFeedBack Use Case can be started asynchronously through this interface by activating the video request link.

ResponsePlot is unit step response of the controlled system.

VideoInterface is an interface for video feedback.

RegistrationControl and WelcomePageControl classes are needed for the collaboration.

Session, Student, ExperimentGenerator and VideoSession entities are used in this collaboration.

This collaboration is more complex. Because ExperimentWithVideoFeedBack Use Case is extended, its collaboration is combined with SendExperimentParameters Use Case.

Sequence of the operations:

1. wantsToDoExperiment: Students start the experiment interface by activating the DoTheExperiment link of the InterfaceC.
2. startExperiment: UserInterfaceC notifies the RegistrationControl object to start the experiment process.
3. getStudentID : Session object gives the StudentID.
4. generateSystemParameters: Emulator generates the plot of the response according to randomised system parameters.

5.1. recordSystemParameters : Generated parameters are recorded to the Student enetity.

5.2.1. startExpInterface: RegistrationControl starts the ExperimentInterface.

5.2.2.a.1. entersPIDParameters: Student enters the controller parameters.

5.2.2.a.2. getPID: Control object gets the control parameters.

5.2.2.a.3. generateOutput: Using student's PID parameters, emulator generates the unit step response of the controller system.

5.2.2.a.4.1.1. recordPIDParametersAndGrade: Student's PID parameters and grade given according to these parameters are recorded to the Student entity.

5.2.2.a.4.1.2. displayOutput: Emulator displays the plot of the response.

5.2.2.a.4.2.1. startVideoRecord: Control object notifies the Session object for video recording.

5.2.2.a.4.2.2. startRecord: VideoSession is opened for recording the video image of experiment environment. Later, instructor can view this file or it can be sent to the student via e-mail.

5.2.2.b.1. streamingVideoRequest: Student can also request a real-time visual feedback asynchronously. This request extends this use case to the ExperimentWithVideoFeedBack use case. In order to make easy to follow the extended path, instead of "a", we add "b" to the message numbers.

5.2.2.b.2. getVideoRequest: VideorRequest is sent to Control object.

5.2.2.b.3. getIPAddr: RegistrationControl object gets the IP address of the student
in order to use in video transfer.

5.2.2.b.4. sendVideoRequest: RegistrationControl object notifies the Session object.

5.2.2.b.5. streamingVideo: Session object starts the VideoSession.

5.2.2.b.6. startStreamingVideoInterface: VideoSession starts streaming video interface.

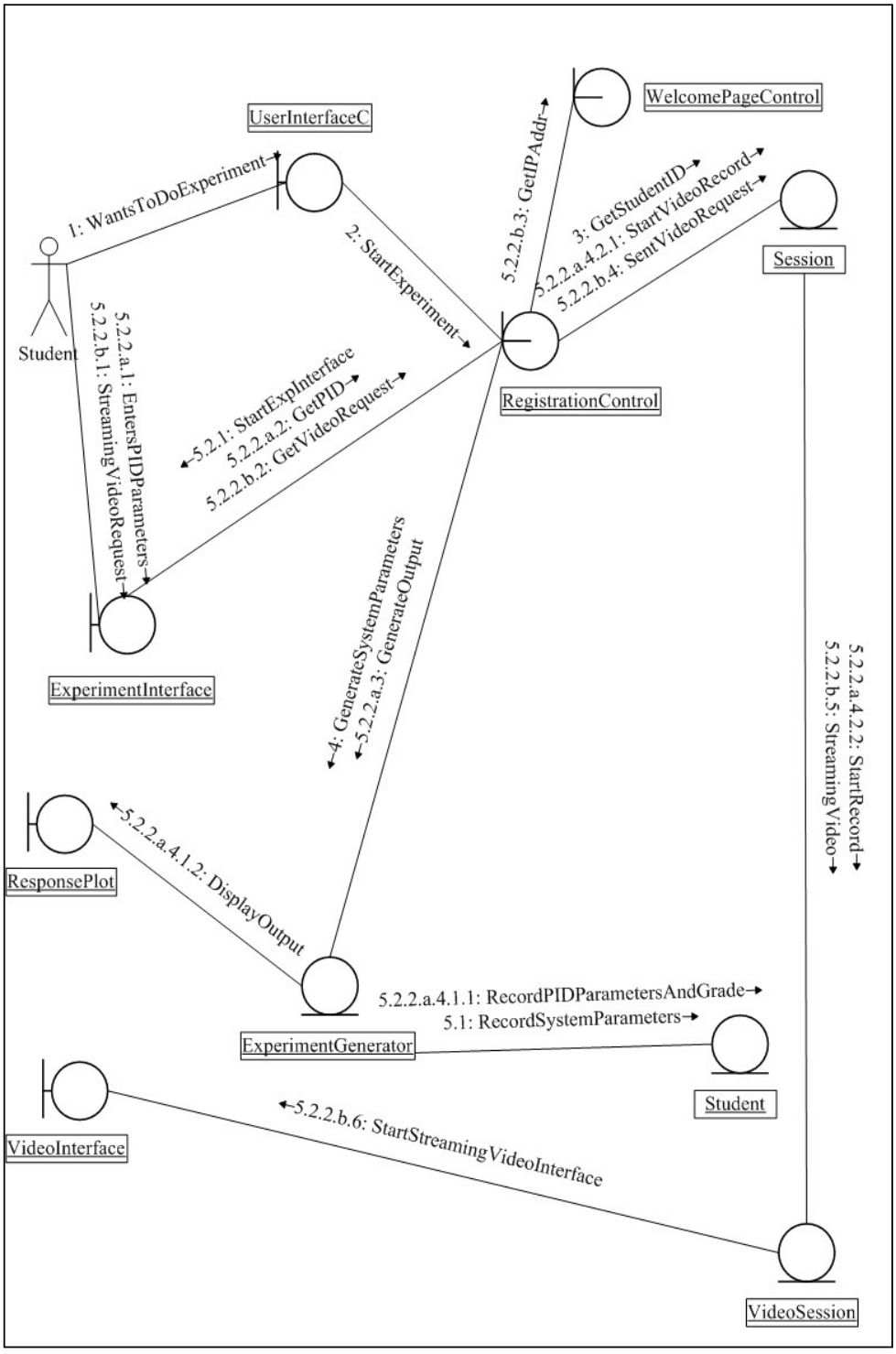

Figure 4.7. Collaboration Diagram for the Send Experiment Parameters Use Case

Henceforth, our actor is the instructor. For all instructor operations, instructor's main menu is used and all use cases related to instructor, starts with the activation of any component on this main menu.

4.2.13 View Registrations Use Case

In this Use case Instructor can view all attributes of the Student entity.

4.2.14 Collaboration Diagram for View Registrations Use Case

This Collaboration is very simple. It has three classes. Instructor makes some queries about students. StudentInterface has necessary text fields for the query.

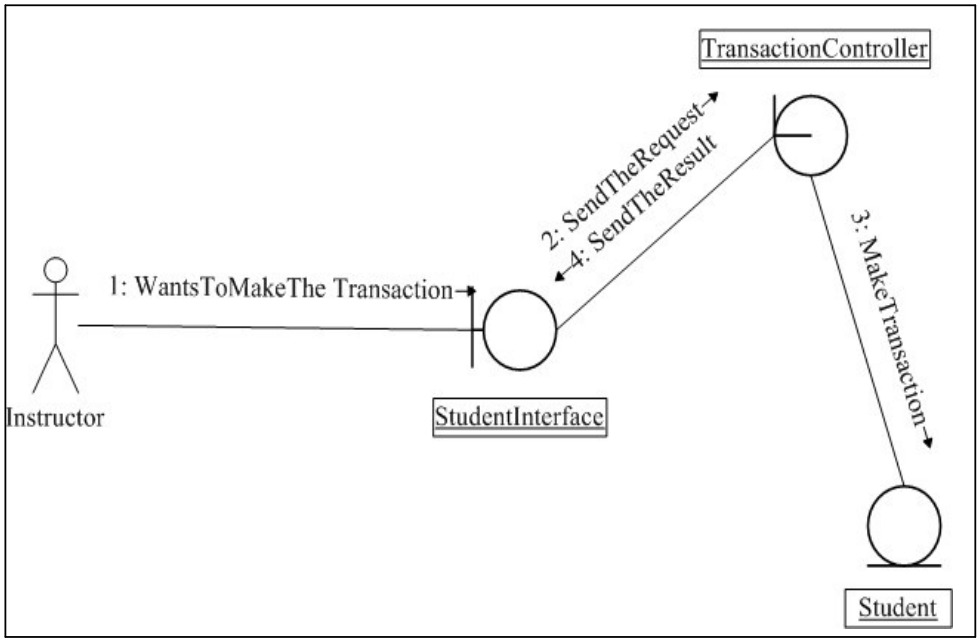

Figure 4.8. Collaboration Diagram for View Registrations Use Case

4.2.15 Update Experiment Schedule Use Case

Instructor updates the Experiment Schedule. Instructor can only make changes on vacant sessions.

4.2.16 Collaboration Diagram for Update Experiment Schedule Use Case

Collaboration diagram has one interface for schedule. This interface can be started with the activation of UpdateSchedule link of the Instructor's main menu.

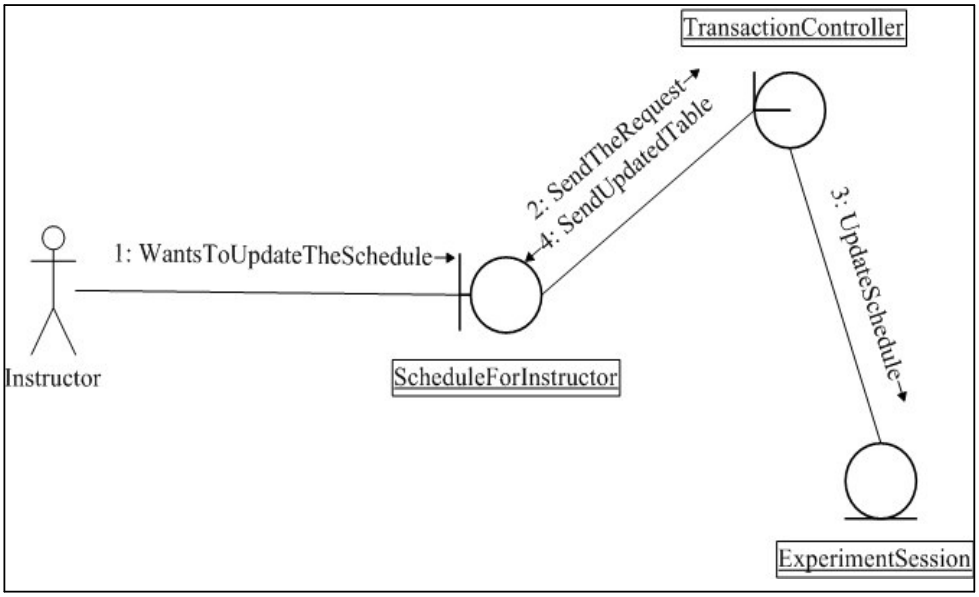

Figure 4.9. Collaboration Diagram for Update Experiment Schedule Use Case

4.3 Class Diagram

Class Diagram is composed of entity classes that are relating with one other entity class. Possible attributes are shown in the diagram. Detailed class diagram can be constructed during the design phase.

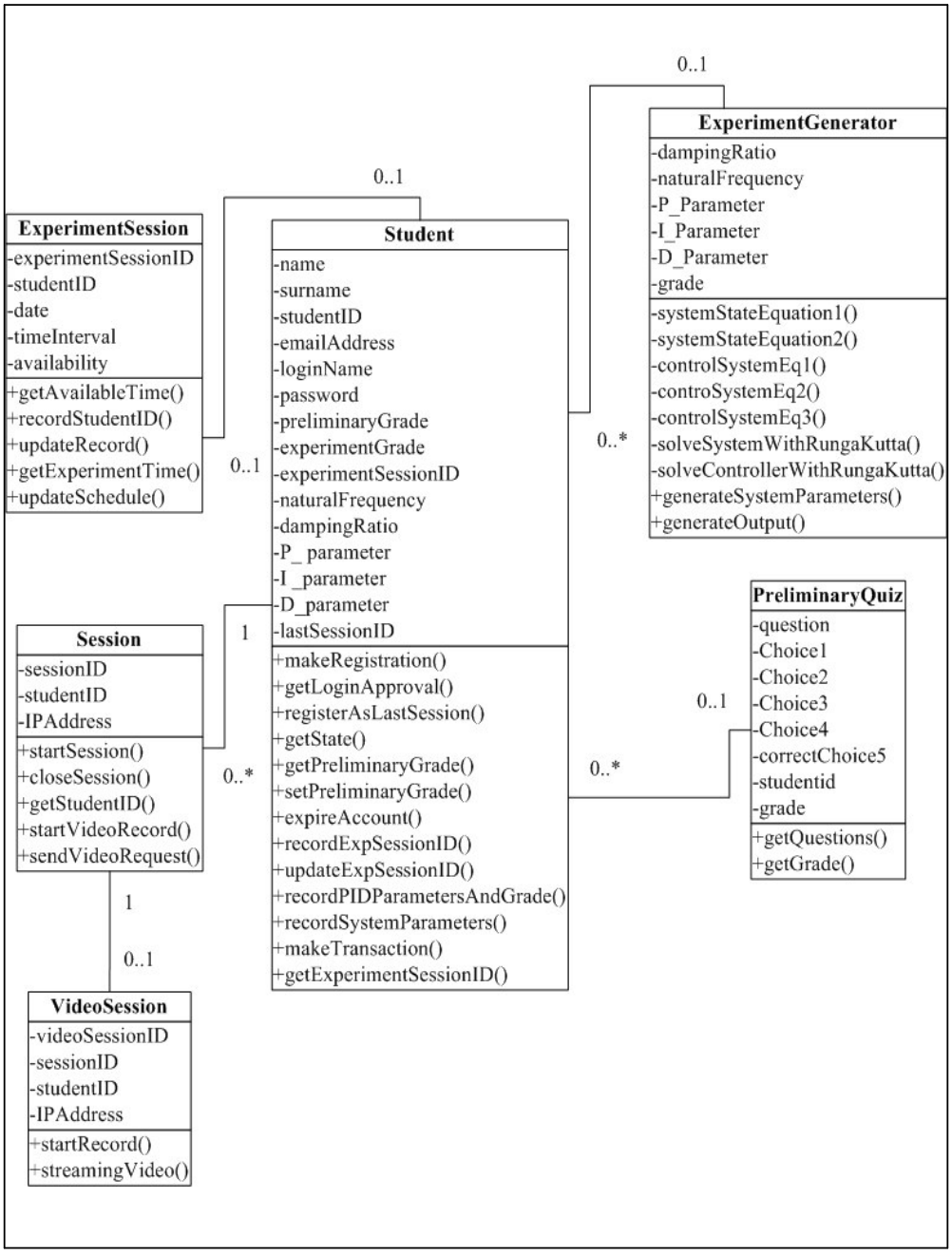

Figure 4.10. Class Diagram of the analysis

Chapter 5

CONCLUSION AND FUTUREWORK

Due to the limited number of instructor and education equipment, distant education methods have been evolved as an alternative against classical education methods. But, still there is a big gap between those new methods and traditional methods of education that require laboratory work. In this thesis, we have tried to fix the needs and propose system architecture in order to reduce this gap. Object oriented analysis of this system has been done and some ideas for Java implementation have been developed throughout the thesis.

As the first step of the system development, an emulator system was designed and implemented using control theory principles. As a future work, firstly, software development process should be completed using the analysis in the thesis. Then, a data acquisition hardware should be obtained for PC based controller implementation of a real plant. The remote access mechanism developed in this thesis can be used for “distance learning” as well as many different industry fields.

REFERENCES

Aktan B., Bohus C. A., Crowl L.A., Shor M.H. (1996): "Distance Learning Applied to Control Engineering Laboratories", IEEE Transactions on Education, 39(3), 320-326.

Alhalabi B., Anandapuram S., Hamza K. (1998): "Real Laboratories: an innovative rejoinder to the complexities of distance learning", The Bulletin of the International Council for Open and Distance Education, vol.2.

Axelson Jan (1999): "Parallel Port Complete", Third Edition Published by Lakeview Research

Axelson Jan (1997): "8051 The Microcontroller Idea Book", Second Edition Published by Lakeview Research

Bennett Simon, McRobb Steve, Farmer Ray (2002): "Object-Oriented Systems Analysis and Design Using UML", Second Edition Mc-Graw Hill Education

Bennett Simon, Skelton John, Lunn Ken (2001): "UML", Mc-Graw International Limited

CommAPI (2002): "Java(tm) Communications API Documentation", www.java.sun.com

Gümüşkaya Haluk (1999): "Mikroişlemciler ve Bilgisayarlar", 1. Baskı Alfa Basım Yayım

Hirumi A., Bermudez A. (1996): "Interactivity, Distance Education and Instructional System Design Converge on the Information Superhighway", Journal of Research on Computing in Education, vol. 29, no 1.

JMFAPI (1999): "Java Media Framework API Documentation", www.java.sun.com

Karagöz İrfan (2001): "Sayısal Analiz ve Mühendislik Uygulamaları", 1. Baskı Uludağ Üniversitesi Güçlendirme Vakfı Yayınları

Kuo Benjamin (1991): "Automatic Control Systems", Sixth Edition Prentice Hall International Editions, New Jersey 07632.

MTS-86C : “MTS-86C 8086 Experiment Setup Guide”, Published by MFG. Co. LTD Taipei, Taiwan.

Ogata Katsuhiko (1990): “Modern Control Engineering”, Second Edition Prentice Hall International Editions, New Jersey 07632.

Pekgöz Numan (2002) : “Java Server Pages”, 1.Baskı Pusula Yayınları

RATL (2003): “Remote Access to Teaching Laboratories Project at University of Western Australia”, http://www.mech.uwa.edu.au/jpt/tele/

RELB (2003): “Remote Experiment Lab Project at the Federal University of Santa Carina”, http://www.inf.ufsc.br/~jbosco/lexrem1i.htm

Sarıoğlu Kemal (1999): “Otomatik Kontrol 2”, 1. Baskı Birsen Yayınevi

Schildt Herbert (2001): “Java 2 The Complete Reference”, Fourth Edition The Mc- Graw Hill Companies

Schimid C. (1999): “A Remote Laboratory Using Virtual Reality on the Web”, Simulation, 73(1), 13-21.