Remote Collaboration in WebLab – an Online Laboratory

by

Victor Chang

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degrees of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING AND
MATER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 23, 2001

Copyright 2001. Victor Chang. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and distribute publicly paper and electronic copies of this thesis and to grant others the right to do so.

Author __________________________
Department of Electrical Engineering and Computer Science
May 23, 2001

Certified by ________________________________
Jesus del Alamo
Thesis Supervisor

Approved by ________________________________
Arthur C. Smith
Chairman, Department Committee on Graduate Thesis
Remote Collaboration in WebLab – an Online Laboratory
by
Victor Chang
Submitted to the
Department of Electrical Engineering and Computer Science

May 16, 1986

In Partial Fulfillment of the Requirements for the Degrees of Bachelor of Science in
Computer Science and Engineering and Master of Engineering in Electrical Engineering
and Computer Science

Abstract

WebLab, an online educational laboratory, lacks features that enable collaboration. We believe that collaborative and tutoring features would enhance the educational value of the system. An investigation into how collaborative features should be added to WebLab is presented. We analyze the groupware design issues of degree of awareness, synchronous vs. asynchronous, and role definition and result in a conceptual design for collaborative WebLab. The “JS prototype” is an initial implementation of our design based on Java servlet technology. In this prototype, a Java servlet acts as a collaboration server that connects with users and routes information among them using a thirteen-command protocol. Each user is given an individual domain and they all share a group domain. In this way, individual and group work can proceed simultaneously but separately. A token scheme is introduced to enforce one-user-at-a-time modification of shared objects in the group domain. We tested the JS prototype in preliminary collaboration experiments. Results from the experiments suggest that collaboration and tutoring increase students’ accuracy. Surveys given to experiment participants resulted in very positive feedback on the JS prototype, verifying the quality of our groupware design.
# 1. INTRODUCTION

1.1 OVERVIEW .......................................................................................................... 5
1.2 RELATED WORK ................................................................................................. 7
1.3 OUTLINE OF THIS THESIS .............................................................................. 11

# 2. THE WEUBLAB SYSTEM

2.1 THE MICROELECTRONICS LABORATORY ....................................................... 13
2.2 THE WEB COMPONENTS ............................................................................... 15
2.3 FUNCTIONALITY .............................................................................................. 17
2.4 COMMUNICATION BETWEEN THE JAVA APPLET AND THE WEUBLAB SERVER .................................................................................................. 18
2.5 CHAPTER SUMMARY ...................................................................................... 20

# 3. GOALS AND ISSUES BEHIND THE DESIGN OF COLLABORATIVE WEUBLAB

3.1 DESIGN GOALS ................................................................................................. 22
3.2 DESIGN ISSUES ............................................................................................... 23
  3.2.1 Degree of Awareness ............................................................................... 24
  3.2.2 Synchronous vs. Asynchronous ................................................................... 27
  3.2.3 Role Definition .......................................................................................... 28
3.3 CHAPTER SUMMARY ...................................................................................... 31

# 4. THE JAVA SERVLET PROTOTYPE

4.1 COMMUNICATION ARCHITECTURE AND PROTOCOL ........................................ 34
  4.1.1 Connecting the Users ............................................................................... 35
  4.1.2 Routing Information between the Users ..................................................... 37
4.2 INDIVIDUAL AND GROUP DOMAINS .................................................................. 39
4.3 THE TOKEN SCHEME ...................................................................................... 41
4.3 USER COLLABORATIVE FEATURES .................................................................. 43
  4.3.1 Individual HP4155 Frame ......................................................................... 43
  4.3.2 Connected Users List .................................................................................. 44
  4.3.3 Group HP4155 Frame ............................................................................... 45
  4.3.4 A Sample Interaction of a Two-Member Group ......................................... 47
4.4 CHAPTER SUMMARY ...................................................................................... 48

# 5. COLLABORATION EXPERIMENTS

5.1 EXPERIMENT METHOD .................................................................................... 49
5.2 EXPERIMENT RESULTS .................................................................................. 51
5.3 RESULTS ANALYSIS ....................................................................................... 53
5.4 SURVEY ............................................................................................................ 57

# 6. CONCLUDING REMARKS AND RECOMMENDATIONS FOR FUTURE WORK

APPENDIX A. COMMUNICATION PROTOCOL COMMANDS ..................................... 64
APPENDIX B. WEUBLAB EXERCISE USED IN COLLABORATION EXPERIMENTS .......................................................... 66
APPENDIX C. JS PROTOTYPE SURVEY ............................................................. 67
Chapter 1

Introduction

1.1 Overview

In fields of engineering, collaboration most often occurs in labs. Students' having sufficient lab experience is crucial to a successful engineering education program. Unfortunately lab equipment is expensive, and lab space on university campuses is limited. Constraints in budget and resources often prevent college students from receiving sufficient access to labs. Realizing these conditions, WebLab, a Java-based application, was developed for microelectronics lab projects at MIT.\(^1\) The application allows students to control a set of electronics lab equipment across the Internet then receive and analyze results produced by the equipment. WebLab reduces lab costs, since numerous students can access the lab equipment almost simultaneously. It also cuts down the need for lab space and the logistics of arranging lab use because students can accomplish their lab projects without being located next to the equipment.

WebLab has one major limitation, however. Lab projects in education or industry usually involve the collaboration of a team, but WebLab was designed for each student to work alone, isolated from other students. The lack of collaborative features in WebLab restricts its potential significantly. Using the current version of WebLab, one of two situations occurs. In one situation, students work without collaborating with other

\(^1\) http://i-lab.mit.edu
students. This removes the elements of collective analysis and group discussion from the students’ lab experience and thereby limits the educational value of the lab exercises. In the other possible situation, students who wish to collaborate physically gather around one computer terminal to use WebLab. Such a scenario is also undesirable because it places geographic requirements on students who wish to collaborate – the World Wide Web’s ability to remove physical distance between people is left completely unexplored. These two unfavorable scenarios lead to our study into how collaborative features could be added to WebLab and what educational value these features would have.

Exactly what features enable collaboration? Dr. Mark et al.\(^2\) dissected collaborative technology into two main categories: communication technologies and information sharing technologies. Those two kinds of technologies are what WebLab needs to enable collaboration. Information sharing technologies are those that give one the ability to make other users aware of his activities and the results of those activities. File sharing technology falls under this category. Communication technologies let people who are collaborating discuss their work with each other. Chat, email, or phone calls are examples of this category.

Communication technologies abound in today’s society. Email has long been a means for people to collaborate with each other. For real-time communication, one can easily connect with others around the world through one of the myriad chat programs available on the Internet. On the other hand, information sharing technologies are much more difficult to implement, since they require the well-coordinated transfer of complex

\(^2\) Mark, G., Grudin, J., and Poltrock, S. *Meeting at the Desktop: An Empirical Study of Virtually Collocated Teams*
data types. Therefore, the effective sharing of WebLab-specific information is the focus of our initial prototypes for collaborative WebLab.

We have implemented two prototypes of collaborative WebLab on two development platforms: the first one on the Microsoft Virtual Worlds platform\(^3\) and the other using Java servlet technology. The Virtual Worlds prototype, hereafter denoted by VW prototype, was eventually discarded mainly because the Virtual Worlds platform places too many demands on the client’s computer hardware; this conflicts with our goal of providing maximum accessibility. Nonetheless, we learned important lessons from its design and development. The Java Servlet prototype, hereafter referred to as the JS prototype, shares many of the design components of its Virtual Worlds predecessor and has been chosen to be the basis for future development.

After the JS prototype was completed, we ran a sequence of experiments with MIT students to evaluate the effectiveness of the tool we developed and to observe the students’ behaviors during virtual collaboration. The students enjoyed being able to collaborate and the results affirmed to us the utility of the JS prototype. The results of the experiment shed light on the usefulness of virtual collaboration, and those results are analyzed in this thesis.

1.2 Related Work

In the last decade, commercial and academic organizations have developed numerous groupware applications that could achieve some level of information sharing.

\(^3\) http://www.vworlds.org/
For this discussion, we categorize the existing applications into two genres: general and specific.

The general applications are more ambitious in scope, as they aim to be the collaboration solution to as many tasks as possible – hopefully all tasks. Microsoft's NetMeeting\(^4\) and Sun's SunForum\(^5\) are two of the most popular groupware programs, and they also offer the most general capabilities. The key information sharing concept in these two programs is *application sharing*. With application sharing, a single instance of an application can be held on the server and be controlled and viewed by multiple users on client machines. On both NetMeeting and SunForum the T.120 protocol for multimedia data communication\(^6\) is utilized to achieve application sharing. Even though the groupware applications themselves are not platform-independent – SunForum is designed for Solaris workstations and NetMeeting can only be installed on Windows PCs – the different applications could work together through the T.120 protocol. Using the T.120 protocol, audio, video, and applications data can be shared between clients of NetMeeting, SunForum, and other applications that accept the protocol. Collaboration in a heterogeneous network environment using these groupware applications is thereby feasible, even though not effortless to achieve.

Since application sharing does not place a requirement on the type of application used in collaboration, it certainly enables collaboration in a broad range of tasks. But the general groupware programs have some significant disadvantages. First, application sharing allows only one instance of the application to operate at any moment. In certain collaboration scenarios, each user needs to have a copy of the application. For example,

\(^4\) http://www.microsoft.com/windows/netmeeting/
\(^5\) http://www.sun.com/desktop/products/software/sunforum/
\(^6\) http://www.lotus.com/products/sametime.nsf/standards/8DDB25B6C08E70E5852566640072FCD2
in most class assignments, students are allowed to collaborate but are required to submit their own solutions in their own words. They would not be able to loosely collaborate in such a way using application sharing. Second, application sharing creates unnecessary network traffic. To achieve the broad applicability of application sharing, a groupware program cannot anticipate the nature of the inputs and outputs of the applications being shared; therefore, it must transfer all inputs and outputs to all users. In a collaborative setting, many of the inputs and outputs to the application do not need to be broadcasted to users. For instance, when a group collaborates on a Microsoft Word document, it is not necessary to notify all users when one of them has moved the document window across the desktop. By transferring all inputs and outputs, excessive network traffic is created. High network traffic inherently places requirements on the users’ connection bandwidth and their geographic distances from the server.

Others have developed general groupware without including application sharing. TeamRoom\textsuperscript{7} exemplifies this type of groupware. The TeamRoom software provides a public environment on its server through which users can exchange messages and post files in a way similar to a team conference room in a real office. Microsoft Network Communities offers similar features over the World Wide Web. While these programs are easily accessible, they can only offer lightweight information exchange and have limited utility in collaborative tasks that involve the substantial use of computer applications.

When one needs a groupware for a task that has specific requirements that are not met by the general groupware solutions, specific groupware applications must be

\textsuperscript{7} Roseman, S. and Greenberg, S. \textit{TeamRooms: Network Places for Collaboration}. 

developed to accomplish the task. Educational Fusion\textsuperscript{8} is a groupware developed for collaborative and interactive computer algorithm education over the World Wide Web. TechTalk\textsuperscript{9} enables students to collaborate on Maple/MATLAB projects online. Specific applications are designed to communicate very particular types of data and utilize the data in specialized ways. These applications offer the obvious benefit of being customized to the specific tasks for which they are intended, but their development requires substantial resources.

Due to the enormous resources required in the development of a specific groupware application, general groupware solutions, such as NetMeeting, are used for most casual or spontaneous collaborative tasks. These general solutions, however, do not fit WebLab's needs in three areas. First, WebLab collaboration requires the transfer of data with specific semantics that cannot be captured by general application sharing groupware programs. Second, we want WebLab users to have access to collaborative features wherever WebLab is accessible, which means every Java-enabled web browser. Solutions such as NetMeeting require installations on client machines, which would limit the mobility of users who wish to collaborate. Third, an effective groupware for students needs a good system for group management, a feature that is missing from available general solutions. For these reasons, we have decided to develop a specific solution for collaboration in WebLab and integrate it into the application.

\textsuperscript{8} Nathan D. T. Boyd. A Platform for Distributed Learning and Teaching of Algorithmic Concepts.

\textsuperscript{9} Johnson et al.. Virtual Office Hours using TechTalk, a Web-Based Mathematical Collaboration Tool.
1.3 Outline of This Thesis

Before we delve into the design of the collaborative JS prototype, we examine the original WebLab system in Chapter 2. Once this necessary background is presented, we proceed to discuss the general design issues behind the JS prototype in Chapter 3. Chapter 4 describes in detail the JS prototype. Chapter 5 describes the educational experiments carried out with the JS prototype. Finally, we end with concluding remarks in Chapter 6.
Chapter 2

The WebLab System

The WebLab system began as a non-collaborative application where each user works in isolation from other users. The JS prototype adds collaborative capability to the system, but its functionality and user interface are mostly left unchanged from the original system. In order to understand how the prototype works, one needs to become familiar with the original, single-user system. This chapter provides a general summary of the original system that is illustrated in Figure 1. The summary sets the foundation for the later discussion of the collaborative prototype.
2.1 The Microelectronics Laboratory

WebLab's purpose is to increase the utility of an array of lab equipment by providing users online access to the equipment. The hardware portion of the WebLab system consists of the devices under test and the instruments that carry out the measurements. The WebLab system, in its current configuration, supports up to eight semiconductor devices, such as transistors and diodes. These semiconductor devices – referred to as Devices Under Test (DUT) – are the objects of observation in student lab assignments (see Figure 4). To aid the students in understanding device characteristics,
the assignments ask the students to observe the responses of the devices when stimulated by various current and voltage signals. The analytical instruments in the WebLab system include a Hewlett Packard HP 4155B semiconductor parameter analyzer and a Hewlett Packard HP E5250A switching matrix (see Figure 2). These two instruments collectively enable the students to select a DUT, send signals to it, and observe its voltage and current response to the signals. The analyzer outputs test currents and voltages through its ports to the selected DUTs, measures the response produced by the connected device, and then graphs the results. The switching matrix is the element that links the selected DUT and the analyzer together and routes signals between them. It serves as a signal routing switch between the DUTs and the semiconductor parameter analyzer.

The HP 4155B semiconductor parameter analyzer has eight bi-directional ports: four SMU (Semiconductor Measuring Units) channels, two VMU (Voltage Measuring Units) channels, and two VSU (Voltage Source Units) channels. In the current configuration, we only connect the four SMU channels to the switching matrix. Eight sets of ports are available on the switching matrix for connection to the DUTs, allowing up to eight devices to be included in the system. The switching matrix routes signals between the semiconductor parameter analyzer and the connected DUTs, so the four SMU ports on the semiconductor parameter analyzer can be linked to any of the connected DUTs. Figure 3 shows how a NPN bipolar transistor is connected to three of the four SMU channels on the semiconductor parameter analyzer.
2.2 The Web Components

The microelectronics laboratory just described represents a standard microelectronics devices characterization lab; in WebLab, however, a few pieces of software and hardware accessories cooperate to make the laboratory a Web-accessible-Lab. These components form a conduit of information between the laboratory equipment and any user in the World Wide Web that has a Java-enabled web browser.

Both the parameter analyzer and the switching matrix understand a control language called GPIB. They each have a unique GPIB address to which external control
commands are directed. A computer can therefore precisely direct the complete setup simply by sending appropriate GPIB commands to the instruments.

Five software components running on a Windows 2000 server work together to provide web access to the lab equipment. One component, the Java applet, is downloaded to the user’s client. Using the input fields on the applet (see Figure 2), user’s can enter control data that the applet pushes onto a web server located in the microelectronics laboratory through VBScript method calls. On the web server, the second software component, an Active Server Page receives the data. The ASP relays the commands to the third software component, a series of Dynamically Linked Libraries (DLL’s) written in Visual Basic collectively called the WebLab driver. This driver translates the user’s request into method calls that are accepted by the fourth software component, a commercially produced GPIB driver called VISA. The VISA driver then translates the Visual Basic calls into the destination language, GPIB. Both the switching matrix and the parameter analyzer are connected to the GPIB interface card, and each one is individually recognized by their unique GPIB addresses. The interface card broadcasts GPIB method calls to all the connected GPIB addresses. Through the cooperation of these five components, the user’s control request is thus relayed to the lab equipment. In the background of this information flow, a fifth component, a SQL database, records and provides user information and transaction data by communicating with the ASP.

The lab equipment targeted by the user request processes it upon receiving the broadcast and responds to the GPIB interface card with the results. Through the same path on which the request traveled to reach the lab equipment, the results travel back to
the client machine. Figure 1 describes how the hardware and software components connect with each other.

2.3 Functionality

With the data conduit between the WebLab user and the microelectronics lab, the user could accomplish three major tasks. First, WebLab provides an interface – the HP4155 Frame – for entering measurement parameters that specify the device to be tested, the channels on the device to be stimulated and observed, and the magnitude of the voltage and current to apply to the channels. Figure 3 shows the input fields on the HP4155 Frame. The applet submits these parameters to the microelectronics lab, and the parameter analyzer performs the test according to the user’s specification. The results of the test are sent back to the applet.

*Viewing the test results* is the second task the user can perform on WebLab. Upon receiving test results, the applet opens a Graph Frame and conveys the results to the user in a graph similar to the one shown in Figure 4. The user can manipulate the axis and view the same set of results in various perspectives. The results can also be viewed in their numerical form or exported into a local file on the client machine.

The third task enabled by WebLab is *preserving the setup*. Each user has allocated storage space in the SQL database on the WebLab server on which setup information can be pushed to for central storage. Since the setups are stored in the database, they can be accessed from any computer terminal, giving the user location-transparent access.
WebLab has been in use by MIT students for more than three years already. More than 100 students out of MIT and Singapore have used the system since its introduction. It has been proven adequate for a variety of semiconductor device characterization experiments. The basic functionalities of expensive, state-of-the-art lab equipment have been provided to students at almost no marginal cost. WebLab has greatly enhanced the utility of the equipment by providing access to students from anywhere, at anytime.

2.4 Communication between the Java Applet and the WebLab Server

The means of communication between the Java applet and the WebLab server deserves particular attention because a similar mechanism enables information sharing in the JS prototype. All communication between the applet and the server is established
through one Java function in the applet called `getPage` that sends HTTP requests from the Java applet to the ASP on the server and returns results.

The applet’s communication with the ASP uses HTTP because it’s a convenient, well-defined protocol for client-server connections. However, in HTTP, the client-server connection lasts through only one request-response cycle before the server terminates the connection. In the JS prototype, HTTP is not used for communication between the applet and the collaboration server because a persistent client-server connection is required to achieve constant bi-directional communication (see Chapter 4 for a detailed description of the communication protocol in the JS prototype).

In WebLab, the applet communicates with the server when the user runs a measurement, saves a setup to the database, or loads a setup from the database. In the first two situations, the applet needs to submit to the server a complete description of the setup on the applet’s Channel Definition panel. The Java function, `getQueryString`, in the applet gathers this setup information into a setup query string. Each field on the HP4155 Frame has a corresponding field in the query string that includes a descriptor of the field and its current value. “&s” are used as separators between the fields on the query string. Here’s an example of the query strings produced by `getQueryString`:

```
username=vtest0&mode=SWEEP&SMU1VName=V1&SMU1IName=I1& …
```

Not all setups can be submitted to the HP 4155B semiconductor parameter analyzer. Certain setups would fail the analyzer’s constraints and raise errors; certain setups would cause damage to the DUT if carried out. It is desirable to submit only legitimate setups to the server because every illegitimate request submitted to the server consumes server resources, and an error message from the server has to travel back
through the Internet to the client, adding to network traffic and wasting user’s time. Therefore, `getQueryString` checks the values it collects against any known constraints on them. If any value fails the check, the function aborts the query string construction and raises an error message that prompts the user to correct the value. The construction of the query string only completes when all the values meet their constraints.

A query string as described above includes all the data necessary to describe a WebLab setup. When the user requests to save a setup, the setup query string is the only data submitted to the server and stored on the database. When the user loads a setup, the stored query string that corresponds to the requested setup is loaded from the database to the applet, and the setup is reconstructed from the query string. To run a measurement, the applet also submits a setup query string to the server. A procedure on the ASP parses the query string into a data structure understood by the WebLab driver and submits it to the WebLab driver that relays the data to the semiconductor parameter analyzer. The procedure then retrieves the measurement results and returns them to the applet in the form of an ampersand-separated query string before closing the connection.

HTTP requests establish the path for communication between the Java applet and the WebLab server, and the query strings convey the content of the communication. This protocol is simple but effective. The JS prototype builds upon the existing protocol to achieve information sharing between WebLab users.

### 2.5 Chapter Summary

It is necessary to understand the WebLab system before one proceeds to explore the collaborative features built on top of the system. This chapter presents a discussion of
WebLab’s hardware and software components and the architecture that brings them together. In the following chapter, we begin to investigate how collaboration can be enabled in this system.
Chapter 3

Goals and Issues behind the Design of Collaborative WebLab

Groupware applications abound today in huge varieties; however, well-accepted guidelines for groupware design do not yet exist. One reason for this is that the field of Computer Supported Cooperative Work (CSCW) is still a young, fledgling area with slightly more than a decade of age. Another reason is that the huge diversity of collaborative tasks makes the development of a set of general design principles a difficult, daunting enterprise. Without consistent guidelines, groupware designers generally derive their own design principles around their design goals. It is therefore especially important in groupware design to set clear goals in the very beginning.

The design goals for the collaborative WebLab prototype are listed in the beginning of this chapter. Then, we present a discussion of the design issues that were considered in the prototype and how they are reflected in the eventual design. Some references to how others have approached these issues are interleaved throughout the discussion to exemplify the relevance of these issues.

3.1 Design Goals

Three high-level goals guide every design decision of collaborative WebLab. These goals are:
- **Enable effective sharing of WebLab-related information between users.** The users who collaborate on WebLab should have the ability to share enough information with each other that they can easily become aware of their partners progress.

- **Provide an environment in which WebLab users can collaborate with minimal conflict and confusion.** In a virtual collaboration setting, clear communication is more difficult than in a physical collaboration setting. Confusion and conflict become more difficult to manage. We must minimize them to avoid collaboration time wasted in confusion clarification and conflict resolution.

- **Maximize the accessibility of the application.** The main benefit of WebLab lies in its ability to bring the lab equipment outside of the physical laboratory. Wide accessibility was, therefore, a major goal of WebLab itself. Collaborative WebLab must be accessible to any user that can currently access the WebLab system.

- **Preserve all the current functionalities of WebLab.** The collaborative features are an extension to the existing WebLab system. All the tasks that individual users can perform on the existing system must still be feasible in collaborative WebLab.

### 3.2 Design Issues

In the design of collaborative WebLab, three general issues are analyzed: *degree of awareness, synchronous vs. asynchronous,* and *role definition.* Based on our research
of related groupware designs and our design goals, we believe that these three issues are most relevant to our design. Each issue is discussed in the sections below.

These issues are discussed with a focus on information sharing, since that is the first priority of collaborative WebLab. Communication technologies are certainly necessary to realize collaboration in WebLab, but we do not discuss the design of such technologies in these sections because we have not yet integrated communication technologies into the JS prototype.

3.2.1 Degree of Awareness

Providing group awareness is perhaps the most important goal of groupware. In an individual working environment, the user creates, edits, and manipulates objects on his computer, but users of other computers cannot observe those activities without some kind of hacking. The user's isolation is a desirable condition when he intends to work alone. The computer protects his work from the subversion and espionage of other people. For example, if the user were a writer authoring a new book on his desktop computer, he would not want other people to be able to plagiarize from his computer. However, the isolation is definitely undesirable if collaboration is needed. In a collaborative environment, users need to be aware of portions of activities occurring on other computers. Reusing the example of the author, what if the writer is coauthoring the book with two other colleagues? He still wouldn't want most people to have access to his document, but he would want his colleagues to have access so they can observe his progress. In this scenario, group awareness becomes essential. A groupware must enable
all the authors to be aware of, to some degree, the activities on their colleagues’ computers.

Solutions that allow group awareness have existed for a long time. In Intranet systems, a user can share directories to other users in his workgroup or domain. On the WWW, a user can share directories to everyone on the Internet. Email attachments are commonly used as an ad hoc means to transfer files between different computers. All these tools of connectivity enable groups working together to have awareness of each other, but they require too much work from the users themselves. A well-designed groupware should give the user the group awareness he needs – and only the degree he needs – without the user having to do much work. Hayashi et al. specifically focus on enhancing group awareness in their design of the Interlocus system\textsuperscript{10}. They believe that a program designer may not be able to predict at program-time exactly how much awareness a group needs, so they give Interlocus the ability to dynamically provide awareness at run-time, based on the monitored interactions between the group members. The individual, therefore, does not have to do any extra work to provide awareness to his group members.

Occasionally, too much group awareness provided by the groupware may be undesirable because it can detract from the individual’s own working ability. From the individual’s perspective, the less awareness he gives to the group, the more control he has over his own work. But, from the group’s perspective, insufficient group awareness could reduce the potential effectiveness of collaboration. Gutwin and Greenberg analyze the tradeoff between individual control and group awareness from three aspects:

\textsuperscript{10} Hayashi, K. et al., *Activity Awareness: Framework for sharing knowledge of people, projects, and places*
workspace navigation, artifact manipulation, and view representation\textsuperscript{11}. They conclude that in all three situations, it is possible to give the individual the control he needs over his software and still provide his group members adequate awareness of his work; but careful design is necessary. How much awareness to give the group, and by what means the awareness is provided are the first issues to consider in the design of a groupware.

The primary goal of collaborative WebLab is to enable WebLab users to share their work with each other. To accomplish this, our JS prototype provides to users awareness of each other’s measurement parameters and results. No other information is shared between the users, since too much information shared between users creates excessive network traffic and thereby reduces the accessibility of the application. To prevent the group awareness features from diminishing the individual user’s working ability, two separate domains – group domain and individual domain – are given to each user.

It is certainly possible to provide group awareness to a user without introducing a separate group domain. For example, one could envision a frame on each user’s display that integrates the displays of every other user in a split-screen fashion. This frame would provide group awareness to the individual user; however, the design is not scalable. The WebLab interface displays a large amount of information to the individual user so that the user’s screen is quite cluttered with just the display of one user’s information. A split-screen frame would not be able to fit multiple users displays without sacrificing significant content or shrinking the displays to unbearably small sizes.

\textsuperscript{11} Gutwin, C. and Greenberg, S.. \textit{Design for Individuals, Design for Groups: Tradeoffs Between Power and Workspace Awareness}
3.2.2 Synchronous vs. Asynchronous

After making the important decision of how much group awareness to provide, the logical next issue to think about is: how immediate should the awareness be provided? Should an individual’s action be instantaneously transmitted to other group members (synchronous) or only be conveyed occasionally (asynchronous)? In a synchronous system, every group member is aware of the actions of the group at all times. It may appear that synchronous systems are always preferable to asynchronous systems because of the low latency in communication, but in reality, it often depends on the nature of the task at hand. Synchronous systems place higher demand on the programmer and more traffic on the connecting network, so it should be avoided when it is not needed.

We implement in the JS prototype a nearly synchronous design. The design is synchronous in the sense that all the users are connected with each other during collaboration. It is, however, not completely synchronous. The awareness of every group member is only synchronized when certain defined events happen during collaboration; therefore, between these events, the members may be out-of-sync. A near-synchronous design was chosen over a completely synchronous design because the operational semantics of WebLab indicate that users only need to be synchronized occasionally. Redundant synchronization only congests the network and complicates the program unnecessarily.
3.2.3 Role Definition

When multiple people collaborate on a task in a physical environment, each group member assumes a separate role. The role could be one that is assigned at the beginning of a collaborative task or one that naturally develops through the interaction among group members. Groups working together in a virtual environment go through the same process – roles would develop during interaction between the group members — but it is more difficult to define roles in a virtual environment. Roles in a group develop through explicit and implicit communication between its members, both of which are harder to achieve in a virtual environment. Therefore, a groupware needs to have an explicit system for assigning roles to users in a group. The most important component of a user’s role is his access rights to objects in the working environment – what he can see and what he can modify. Once that is determined, his access rights may indirectly affect other aspects of himself, such as his appearance in the environment and his view of the environment.

Let’s use the example of a group coauthoring a book once again to clarify the idea of roles. Assume there are two separate roles in the workspace: a writer and an editor. The writer’s role only enables him to contribute to the parts of the book he is responsible for. The editor’s responsibility, on the other hand, includes editing the work submitted by all the writers and putting together the final book. Under this arrangement, each writer has read/write access to his own document, but the editor has read/write access to each writer’s document. To distinguish the editor from the other users in the group, his username is bold typed in the workspace. The editor’s view of the workspace also differs
from a writer's view because of the editor's extra access rights. Each writer's view of the environment only displays his own document, while the supervisor's view shows every writer's document. The editor can therefore read and modify any document he desires.

In designing the role structure for a groupware, one should think about three important issues. First, one needs to decide whether the environment should provide different roles. In applications such as TeamRoom\textsuperscript{12}, all users have the same access rights. In TeamRoom, each user is considered an equal member of a team that can view or post to any object in the workspace. This works well for the task that TeamRoom is designed for: providing a centralized depository of information for a team. However, for many tasks, as in the coauthoring example above, different roles need to exist in a group.

The second issue to consider in role structure design is whether two users can simultaneously assume similar roles, and therefore, concurrently make modifications to objects in their workspace. Allowing concurrency adds parallelism to collaboration, but also introduces a very tricky problem of concurrency resolution. What should happen when two users concurrently modify the same object? Should one user's modification take priority over another, or should they be combined? Researchers, such as Munson and Dewan\textsuperscript{13}, have tried to tackle the problem of controlling concurrency. Concurrency control is not a simple task, so a groupware architect should think carefully before allowing concurrency in an application.

The third issue one should think about is the degree of flexibility in role assignment. Many systems utilize an inflexible role assignment scheme. Each user is assigned a fixed role upon the creation of his account, and the role cannot be modified by

\textsuperscript{12} Roseman, S. and Greenberg, S.. \textit{TeamRooms: Network Places for Collaboration.}

\textsuperscript{13} Munson, J., and Dewan, P.. \textit{A Concurrency Control Framework for Collaborative Systems.}
anyone but the system administrator. However, in many collaborative settings, the roles of individual group members evolve through the collaborative activity. Recent research has developed more flexible role structures. Kansas\textsuperscript{14}, for example, is a shared-space environment that supports dynamically changing roles. In Kansas, an individual’s access rights are determined by his capability set. The set can evolve based on his activities, and it can be passed to other users. A flexible environment as this is more suited for highly interactive, dynamic tasks.

We envision collaboration in WebLab to occur in two scenarios: 1) a group of students collaborating on an assignment and 2) students working with an instructor in tutorial fashion. In both scenarios, users collaborate in a democratic fashion. Conceptually, there does not need to be a differentiation of roles between collaboration users – one role definition for everyone works well. However, this would allow concurrency, a condition with serious negative effects. If every user assumes the same role, all users could modify the group objects simultaneously, and significant conflict and confusion would occur among the group. The management of concurrency also introduces much unnecessary complexity into the program logic. To avoid concurrency, we introduce two roles into the JS prototype that are differentiated by a token. When one assumes the role of \textit{token holder}, one obtains the write to modify group objects; otherwise, one could only observe them. The token is implemented not to create a leadership role into WebLab collaboration, but rather to enforce non-concurrency.

WebLab collaboration involves considerable interaction between users, so the assignment of the group token is flexible in our prototype. The token may circulate among a group during a session without restrictions. WebLab collaboration is, however,

\textsuperscript{14} Smith, R.B et al.. \textit{Supporting Flexible Roles in a Shared Space}
not a dynamic task. It involves the repetition of only a few pre-defined tasks, such as preparing a test vector for measurement and configuring a graph of the results; therefore, the prototype does not provide for the definition of new roles at runtime. *Token holder* and *non-token holder* are the only two available roles throughout a collaborative session.

### 3.3 Chapter Summary

We set four design goals for collaborative WebLab and analyzed three general design issues in the context of our goals. From the analysis emerges a conceptual design. The next chapter describes in detail the JS prototype – a preliminary implementation of this conceptual design.
Chapter 4

The Java Servlet Prototype

Effective sharing of WebLab-related information is the focus of our collaborative prototype. Since general communication can be achieved easily through any of the numerous chat programs available, we defer development of a communication tool temporarily. We assume that once a prototype that enables information sharing is available, the users can communicate with each other through a third-party chat program.

To achieve information sharing, the WebLab system needs to include a collaboration server that can route information between WebLab applets on different client machines. Two separate development platforms were considered for the collaboration server for our collaborative prototype of WebLab. Each platform was considered because it provided tools for communicating with the WebLab Java applet. The first platform considered was the Microsoft Virtual Worlds platform, a development environment for 3D virtual-reality worlds served over the World Wide Web. Its server package offered hooks that allowed web browsers to communicate with worlds stored on the server using VBScript. We developed a prototype on Virtual Worlds but eventually ceased the project. The platform required client machines to run the Windows operating system and install web browser plug-ins developed by Microsoft, the 3D graphics places heavy demands on the client machines, and communication between the WebLab Java
applet and the Virtual Worlds server involve hefty marshalling and unmarshalling of COM commands – all of these characteristics restrict the accessibility of the VW prototype without offering significant benefit to our needs.

After rejecting the Virtual Worlds platform because of its excessive features, we decided to develop all the collaborative features from scratch and customize them to WebLab. Since the front-end interface in WebLab was developed in Java, our natural choice was to develop the collaboration back-end in Java also. We, therefore, developed a new prototype that includes a Java servlet as the collaboration server. Compatibility between the WebLab clients and the collaboration server is ensured when the same language is used to implement the client and the server. This chapter describes the prototype that derived from the addition of a Java servlet to WebLab and substantial modifications to the WebLab Java applet. Figure 7 details the architectural layout of the JS prototype.
4.1 Communication Architecture and Protocol

The collaboration server is developed using the Java servlet package included in the Java 2 Platform, Enterprise Edition (J2EE). It is served to the World Wide Web through the open-source Tomcat 3.2 servlet server introduced by Apache. It must reside on the WebLab server, since Java applets can only communicate with the server from which they are served; but functionally, the collaboration server is separate from the other components on the WebLab server. The collaboration server performs the crucial tasks of connecting the users and routing information between them, and these tasks are
achieved through a communication protocol of thirteen commands between the WebLab applet and the collaboration servlet.

The commands are wrapped in ampersand-separated command query strings similar to the setup query strings used in communication between the WebLab applet and the ASP on the WebLab server. Here is an example command query string from the collaboration server to the applet that passes the \texttt{CONNECTED\_LIST} command with its two arguments, "vchang" and "vtest1":

\texttt{CONNECTED\_LIST:vchang&vtest1&}

Using this simple command protocol, the collaboration server connects WebLab users and opens a communication channel between them. The following two sections describe how the communication protocol works. When relevant, specific commands are mentioned; although, not all thirteen commands of the protocol are covered in the following sections (see Appendix A for the complete set of commands).

4.1.1 Connecting the Users

In the original, single-user version of WebLab, users cannot collaborate because they lack a means of communication between them. In the JS prototype, the collaboration server serves as a central communication center. Once initiated, the collaboration server listens to the port designated to it for socket connection requests from any WebLab applet. An applet that desires to connect begins by creating a thread that sends a socket connection request from his WebLab applet to the collaboration server. The server spawns a new thread to handle the request and to establish the socket connection.
It is important to note that HTTP is not used in this communication despite the advantages of having well-defined provisions for handling HTTP in the Java servlet package. HTTP is not used because under the protocol, a connection between the client and the server persists through only one request and response, after which the server closes the connection. This works well when the client always initiates communication and pulls data from the server. However, a collaboration server often needs to initiate communication with specific clients and push data to them. In order for pushing to occur using HTTP, clients have to repeatedly poll the server to check for data. Polling is inefficient unless it occurs at high frequency, but frequent polling occupies computer resources and adds unnecessary network traffic. A persistent socket connection is much more effective as a bi-directional communication channel between client and server. As Figure 7 shows, the WebLab applet communicates with the ASP using HTTP connections, but it communicates with the collaboration server on a persistent socket connection.

Once a persistent socket connection is established between the WebLab applet and the collaboration server, they both listen continuously on the connection for incoming commands. The applet sends a JOIN command to the server, which stores the socket and the corresponding username into its list of connected users and pushes a CONNECTED_LIST command back to the newly connected applet along with the list of connected users. After this series of exchanges, the collaboration server becomes aware of the connecting user, and the user gains awareness of all the users connected to the collaboration server.
4.1.2 Routing Information between the Users

Once the communication channel is established as just described, a connected applet can send commands to the collaboration server and receive commands from the collaboration server. Information can be transferred between all the connected applets through these communication channels. When a user wishes to share information with other connected users, he first transmits it from his applet to the collaboration server, and the collaboration server pushes it to the appropriate recipient.

Collaboration in WebLab requires the sharing of measurement parameters and measurement results between users. To share a set of measurement parameters, the user’s applet begins by calling the Java function, `getCurrentInputs`, in the applet. This function does what `getQueryString` does—it gathers all of the current values on the applet’s HP4155 Frame and puts them into a setup query string. The functions, `getCurrentInputs` and `getQueryString`, are very similar, but they differ in two ways. First, unlike `getQueryString`, `getCurrentInputs` does not check the values for their legitimacy because the values are not being submitted to the lab instruments to be measured, they’re merely being shared with the other WebLab users. There is no reason to censor the communication between users. The censorship of the values occurs only when the user plans to submit his setup to the lab instruments. Second, `getCurrentInputs` collects more values than `getQueryString`. `getQueryString` only gathers the setup information on the HP4155 Frame because that information is all that is needed to perform a measurement. However, the query string constructed by `getCurrentInputs` is used for others to reconstruct everything displayed on the user’s screen. Thus, `getCurrentInputs` also gathers the inputs on the graph panel that dictate how the graph is displayed.
After `getCurrentInputs` has constructed the setup query string that represents the user’s inputs, the applet appends the string to the command, `PUSH`, as its argument. The new command query string is then submitted to the collaboration server. Upon receiving the command, the collaboration server stores the setup string as the current representation of the group’s work. Then it appends the setup string to a `PUSH` command and broadcasts the command string to every connected applet. The applets parse the setup string and update their displays with the new setup.

Measurement results are shared between users in a similar fashion. Whenever a measurement is run, the test results are saved on the applet as an ampersand-separated query string. These result query strings are generally very long, depicting every data point in a graph. When a user wishes to share his graph with others, the applet appends the current result query string to a `GRAPH` command and sends the resulting command query string to the collaboration server. The server saves this string and broadcasts it to the other applets, which then refreshes their graphs with the results contained in the string.

So far, we have only described how users can push data to other users, but the collaboration server can also be a source from which to pull data. This is useful to a user whose applets may be out-of-sync with the rest of the group. The applet can submit a `PULL_RESULT` command and a `PULL_INPUTS` command to the collaboration server. When the collaboration server receives these commands, it pushes the stored current results and inputs back to the requesting applet. The applet can then synchronize its display with the rest of the group using the data received.
4.2 Individual and Group Domains

Based on the previous description of the collaboration protocol, one might think that individual users have no control over their applets, as it seems that users may modify each other's applets at will. In the JS prototype, it is true that users may modify each other's applets; but users can also maintain control over their individual work that they do not wish to share with others. These two conflicting conditions can be true simultaneously because two separate domains are implemented in the applet of the JS prototype.

Before connecting to the collaboration server, each user owns two frames in the individual domain: an Individual HP4155 Frame (the frame that contains the channel definitions for the HP 4155B instrument) and an Individual Graph Frame (the frame that displays the graph). Since the user has not yet connected to the collaboration server, no other user has access to his two frames. Figure 7 shows that the collaboration server is a self-contained extension to the WebLab server. If the applet is has not established a connection with the collaboration server, the JS prototype functions exactly the same as the single-user WebLab

When the user logs on to the collaboration server, two frames in the group domain – Group HP4155 Frame and Group Graph Frame – appear. As can be seen in Figures 8-11, the group frames and the individual frames share nearly identical appearances. These two frames have the same WebLab functionality as their counterparts in the individual domain, but they are shared with the other connected users. The holder of the modification token (see Section 4.3) can modify the frames in the group domain by means of the communication protocol. However, the individual domain
remains in the control of the individual user – no other user could have access to this domain. Thus, while the user could stay abreast of the group’s collective progress by observing the frames in the group domain, he could also perform his own experiments in the individual domain without concern for unanticipated modifications by other users.

Data could transfer freely between the individual domain and the group domain. So, if the user wants to perform individual experiments based on the data in the group domain, he could \textit{download} the data in the group domain into the individual domain. On the other hand, if the users wishes to share his individual work with the group, he could obtain the token and \textit{upload} the data in his individual domain into the group domain. Once the data reaches the group domain, they can be pushed to other users.

\textbf{Figure 8 – Individual HP4155 Frame} \hspace{1cm} \textbf{Figure 9 – Group HP4155 Frame}
4.3 The Token Scheme

The communication protocol only provides the means for communication. Beyond that, there needs to be a set of rules regulating the communication – rules that dictate who can communicate, when communication can occur, and what they can communicate. Communication control should occur on two levels: intra-group and inter-group. In its present configuration, the JS prototype can only handle one group of users. All users who connect to the collaboration server automatically get assigned to the default group. Therefore, there exists no need for intra-group control.

The key purpose behind inter-group communication control is the prevention of unwanted concurrent modifications of the group’s objects; in the case of WebLab, the group’s shared HP4155 Frame and Graph Frame. If multiple users attempt to modify the shared objects, they may overwrite each other’s modifications and chaos inevitably
occurs. We introduce a *modification token* to ensure that concurrent modifications do not take place.

The modification token represents the right to modify the group's shared objects. Only one token exists in a group, and therefore, only one user holds the right to modify at any moment. A user can obtain the token by default, by hand-over, or by inheritance. By default, the token is assigned to the first user who logs on to the group. It is naturally undesirable to have one user dictate the group's work throughout an entire collaboration session. The token holder can pass the token to any other user at anytime. If the token holder disconnects from the collaboration server before passing the token to anyone else, a randomly selected user inherits the token.

This token scheme inherently implies optimistic assumptions about the cooperativeness between collaborating users because it does not provide a method for the token to be taken away from a connected token holder. We assume that no user would "hog" the token against the group's objection. Without that assumption, a mechanism such as a group-voting scheme would be necessary to allow the group to "impeach the hog." We also assume that collaborating users would give each user his fair chance to hold the token. As we later discover through tests of the prototype, our assumptions hold in most student groups.

Under the token scheme, only the token holder can push setups and results to the server. On every other user's applet, all the controls that trigger actions that involve such pushing are disabled. When holding the token, the user can freely modify the frames in his group domain and push the data in the group domain to the collaboration server. Any pushed data received from the token holder by the collaboration server is automatically
pushed to every other connected applet, whose group domains would be updated accordingly. If an applet does not hold the token, it can still pull data from the server. Pulling data is allowed because it doesn’t modify any of the shared objects and, therefore, would not conflict with the token holder’s actions.

4.3 User Collaborative features

To the WebLab user, all the collaborative features described in this chapter are encapsulated in a few controls on the Individual HP4155 Frame, the Connected Users List, and the Group Hp4155 Frame. The sections below describe each frame and the collaborative controls on each frame.

4.3.1 Individual HP4155 Frame

The Individual HP4155 Frame (shown in Figure 8) is the starting point of a session in the JS prototype. It has all the features of the original WebLab HP4155 Frame plus a Collaboration Menu with the following options:

- **Log On**

  This action connects the user to the collaboration server. Before logging on, the user works individually, just as in the original WebLab. Once he logs on, the Group Hp4155 Frame and the Connected User list appears. At this time, the applet pulls data from the collaboration server, so that its display is synchronized with the other already connected users.
• **Upload**

  This action is only enabled if the user has the token. This action is the means to transfer information from the individual domain to the group domain. By choosing this action, everything on the Individual Hp4155 Frame and Individual Graph Frame would be uploaded to the Group Hp4155 Frame and Group Graph Frame. The applet then pushes the data in the group domain to the collaboration server, which, in turn, pushes the data to every other user.

• **Download**

  All users have the right to perform this action at any time. When this action is chosen, all the information on the Group Hp4155 Frame and Group Graph Frame is downloaded to the Individual Hp4155 Frame and Individual Graph Frame.

### 4.3.2 Connected Users List

The Connected Users List (shown in Figure 12) is a simple frame whose sole purpose is to provide the user awareness of the other connected users. Immediately after the user logs on to the collaboration server, this frame appears and displays the username of every connected user. As can be seen in Figure 12, one – and only one – username has an * next to it. This denotes that the user corresponding to this username owns the token.

The *Pass Token* button is the only control on this frame. This button is enabled only when the user has the token. The token holder can pass the token to any connected user by selecting the user’s username on this frame and clicking the *Pass Token* button.
Instantly, the * moves next to the new token holder’s username, and the Pass Token button becomes disabled for the user who just passed the token.

![Connected Users List](image)

Figure 12 – Connected Users List

### 4.3.3 Group HP4155 Frame

The Group HP4155 Frame also has a Collaboration menu that includes the following controls:

- **Log Off**

  This action disconnects the user from the collaboration server and ends the collaboration with the group. The group frames close, and the user’s username is taken off the Connected Users List of the remaining connected users. At this point, the user can only work in the individual domain. If a user exits the application without selecting this option, the application automatically disconnects the user from the collaboration server.
• **Show Users List**

  If the user has closed the Connected User’s List, choosing this action opens it again.

• **Synchronize**

  This action does one of two things, depending on whether the user holds the token. If the user holds the token, choosing this action *pushes* everything in his group domain to the collaboration server. The server stores the pushed data and broadcasts them to the other connected users. On the other hand, if the user does not hold the token, this action *pulls* the group setup from the collaboration server and updates the group domain.

Besides the controls in the Collaboration menu, there is one other control that affects the user’s collaborative session. The *Run Measurement* menu option on the Group HP4155 Frame has been modified to trigger an automatic synchronization. This option is only enabled on the token holder’s applet, and when selected, it submits the setup on the Group HP4155 Frame to the WebLab server, graphs the measurement results on the Group Graph Frame, and then pushes everything in the applet’s group domain onto the collaboration server. Thus, the whole group is synchronized after every measurement run in the group domain performed by the token holder.
4.3.4 A Sample Interaction of a Two-Member Group

This section describes a sample interaction that exemplifies how the different components of collaboration come together in the user experience. To begin the interaction, WebLab user, Bob, selects the Log On menu option in the Collaboration Menu on his Individual HP4155 Frame. Upon logging on, his Group Hp4155 Frame and Connected Users list appear on his screen. He sees that Alice is already logged on and possesses the token. He asks Alice (through chat) to show him what she has done.

Alice has already entered a set of inputs and run a measurement in her individual domain. She selects the Upload action in the Collaboration Menu on her Individual Hp4155 Frame, after which all her inputs and the graph of her results get transferred onto her group frames (the Group Hp4155 Frame and the Group Graph Frame). She could perform the upload because she possesses the token.

Bob immediately sees his group frames updated with the information on Alice’s group frames. He likes what he sees, but he would like to experiment with Alice’s results. So, he selects the Download option in the Collaboration Menu on his Individual Hp4155 Frame to transfer all the information on his group frames onto his individual frames. After he has made some modifications on his Individual Hp4155 Frame, he runs the new measurement and receives a new graph on his Individual Graph Frame. He wants Alice to see the new results, so he asks Alice to pass him the token. Alice complies by passing Bob the token through the Connected Users list.

Now that Bob has the token, he could upload the information on his individual frames onto the group frames. He selects the “Upload” action on his Individual Hp4155 Frame to do that. After the information is uploaded, Alice’s group frames are
immediately updated with Bob’s inputs and results. She and Bob, having the same information on their group frames, now intelligently discuss it through chat.

In this sample interaction, neither Alice nor Bob modifies setups or runs measurements directly from the group domain. This is the cleanest method of collaboration because it maintains a clear boundary between the individual and group domains. Nonetheless, the token holder could in fact work directly in the group domain, since no other user could overwrite his work.

4.4 Chapter Summary

The collaboration server is the key component of the JS prototype. It connects with WebLab applets on persistent socket connections and communicates with them in a thirteen-command protocol we defined. Having the collaboration server as a router, applets connected to it can share information with each other. Each applet consists of the *individual* and *group* domains, allowing the user to toggle between individual and group work. To avoid concurrency in the group domain, a strict token scheme is implemented to enforce one-user-at-a-time modification. The following chapter depicts how we used the JS prototype to learn about the effects of collaboration.
Chapter 5

Collaboration Experiments

Once the JS prototype had been completed, we had a groupware application available for preliminary testing. Before further development of the collaborative WebLab, we wished to observe the effects of collaboration on students’ work in WebLab and get feedback on our prototype design. To these ends, we gathered a number of MIT students whom are familiar with WebLab and ran a series of experiments with the JS prototype. These experiments were simple preliminary tests, but the results we obtained from them still revealed some interesting patterns. This chapter details our method of experimentation and discusses the results we obtained.

5.1 Experiment Method

Twenty-one MIT students were recruited for these experiments. All of the students were enrolled in an undergraduate course in microelectronic devices that had been using WebLab for lab assignments. Recruiting all of the participants from students in this course ensured that they were familiar with the use of WebLab. It also reduced potential bias in the results caused by differences in the participants’ backgrounds.
A six-part WebLab exercise was designed for the experiments (See Appendix B). This exercise was moderately difficult and should be feasible by all participants. Eight experiment sessions were performed. Each session was assigned to one of three groups: *individual, collaboration, or tutorial*. During the experiment sessions, the subjects were asked to complete the exercise, and the way in which they were requested to complete the exercise depended on the group to which the experiment session belonged.

One experiment session with six participants was assigned to the *individual* group. Participants in this group were required to complete the WebLab exercise using the single-user version of WebLab, without any collaboration. We measured the number of minutes each participant took to complete the exercise. After the experiments, the participants’ solutions were graded on an objective 60-point scale, and the scores were recorded. This group plays the role of the reference control group that reflects students’ behavior when working alone.

Two experiment sessions of two participants and three experiment sessions of three participants were assigned to the *collaboration* group. Participants of this group were distributed into five separate sessions because the JS prototype could only handle one collaborating group at a time. During each session in this group, the participants were asked to collaborate together on the WebLab exercise and turn in one set of solutions as a group. The JS prototype and a chat program were provided to each of them for information sharing and communication with team members. Their conversation in chat was logged in a time-stamped log. The time taken by the group to complete the exercise and the score it earned on the solution were recorded. At the end of the session, a four-question survey about their collaboration experience was given to the participants.
The results from these sessions would indicate the effects of collaboration on students’ working behavior.

Two experiment sessions of one participant each was designated under the tutorial group. Participants in this group were assigned the WebLab exercise and offered online tutoring by the course instructor who designed the exercise. During each session, the participant was provided the JS prototype and a chat program. The tutor was connected to both collaboration tools and available to answer questions from the participant about the exercise. As in the other groups, the time taken by the participants to complete the exercise and the score they earned were recorded. Their chat conversations were logged, and the survey given to the collaboration group was also given to these participants. The results would demonstrate the effectiveness of online tutoring in WebLab.

5.2 Experiment Results

As described in the previous section, the time taken for each group or individual to complete the assigned exercise and the score of each solution were recorded. The time taken indicates the speed of the group or individual, and the score of the solution reflects the accuracy of the group or individual. We calculated the average of the two metrics for each group (individual, collaboration, and tutorial) and compared them.

Figure 13 shows the comparison of the average time taken between the three groups and their standard deviation (the length of each data segment is two standard deviations). The chart indicates that the participants in the individual group took the least amount of time, and the participants in the collaboration group took the most time.
Further, the average for the collaboration group was significantly higher than the averages of the other two groups, exceeding the individual group’s average by 35 minutes and the tutorial group’s average by 28 minutes. The individual group and the collaboration group have similar standard errors, while the tutorial group has a much larger standard error.

A comparison of the average scores of the three modes reflects a different pattern, as shown in Figure 14. The individual group received the lowest average score, and the tutorial group received the highest. Again, the individual and collaboration groups have similar standard deviations in the score. The tutorial group has zero standard deviation.

![Average Time Taken to Complete Exercise](image)

**Figure 13** – Comparison of average time taken between the three groups and their standard errors.
Figure 14 – Comparison of average score between the three groups and their standard errors. The maximum score is 60.

5.3 Results Analysis

The experiment results suggest that people work at much higher speeds when working alone; but when collaborating with others, they work more slowly but with higher accuracy. What is the reason behind this behavior? Several hypotheses could be applied to explain this, but the logs of the chat conversations that occurred during the experiment sessions reveal a good explanation. The chat logs show that in every collaborative session – collaboration session or tutorial session – a substantial amount of time was spent on mutual confirmation of ideas.

The conversation below, extracted from the chat conversation between a group of three participants, demonstrates the amount of effort spent in mutual confirmation. The group has just begun working on part 3 of the exercise, and they encounter difficulty:
weblabtest2 (4:55:25 PM): now what, i goes through ic and ib
weblabtest1 (4:55:45 PM): I=IE
weblabtest3 (4:55:51 PM): we have a short here
weblabtest3 (4:56:02 PM): its basically just 1 diode i think
weblabtest3 (4:56:12 PM): i=ib
weblabtest3 (4:56:15 PM): there is no IE
weblabtest2 (4:56:29 PM): ie is comm
weblabtest2 (4:56:38 PM): why i=ib and not i=ic
weblabtest1 (4:56:44 PM): Ve is comm
weblabtest2 (4:56:48 PM): cuz it shorts?
weblabtest3 (4:57:01 PM): im isorry, i meant ic=0
weblabtest2 (4:57:05 PM): ok
weblabtest3 (4:57:07 PM): the wire goes straight into b
weblabtest3 (4:57:13 PM): so i=ib
weblabtest2 (4:57:14 PM): i’ll buy that
weblabtest3 (4:57:24 PM): and it hink v=vb as well
weblabtest2 (4:57:33 PM): yep
weblabtest3 (4:57:34 PM): this should just be a diode
weblabtest2 (4:57:59 PM): how’s this different that 1
weblabtest2 (4:58:07 PM): cuz ic=0 instead of common
weblabtest2 (4:58:17 PM): ?
weblabtest3 (4:58:25 PM): i’m not sure, it might be the same thing

As can be seen in the conversation, the group begins by discussing the approach. The members pose questions to each other and seek confirmation of their ideas. Part 3 of the exercise was found to be the most difficult part of the six-part exercise. Each member of the group works on solutions of their own, but none of them jumps to conclusions without confirmation from the group:
After 30 minutes of experimentation, weblabtest1 believes he has a correct solution. He shares the solution with the group. Weblabtest3 agrees with the solution, but weblabtest2 does not agree with it. Thus, the group waits for weblabtest2 to do more testing:

weblabtest1 (5:44:04 PM): ok 2 has the token
weblabtest1 (5:44:34 PM): plot vs -IE?
weblabtest2 (5:44:57 PM): did you get it
weblabtest1 (5:45:06 PM): yeah
weblabtest2 (5:45:13 PM): looks similar
weblabtest3 (5:45:24 PM): ok so is this right?
weblabtest2 (5:45:28 PM): i like it
weblabtest3 (5:45:45 PM): ok, so go with this?
weblabtest1 (5:45:55 PM): did the graph vs just IB+IC look like this?
weblabtest3 (5:46:04 PM): yeah, it did
weblabtest1 (5:46:04 PM): aha
weblabtest3 (5:46:05 PM): weird
weblabtest1 (5:46:08 PM): look at it in linear
weblabtest1 (5:46:13 PM): in log i think it takes absolute?
weblabtest2 (5:46:20 PM): that makes sense
weblabtest3 (5:46:22 PM): ah
After weblabtest2 submits his new results, the group finally reaches a gleeful agreement. The repetitive process of mutual confirmation caused the group to spend 53 minutes on part 3 alone – more than the average time the individual group took to complete the whole exercise. However, this group’s solutions received a score of 59, far higher than the average score in the individual group. Mutual confirmation certainly slows the students down, but it also garners the whole group’s knowledge and increases the accuracy of the group. It explains why the individual group has a higher speed than the collaboration and tutorial groups but achieves lower accuracy than the other groups.

We hypothesize that the tutorial group’s high average score can be attributed to the fact that the participants had a knowledgeable tutor with whom to confirm his ideas. However, since we only ran two tutorial sessions, we do not have sufficient evidence to confirm this claim. The low number of data points might also explain the tutorial group’s large standard deviation in time taken and non-existent standard error in solution score. In fact, with only 6 individual data points and 5 collaboration data points, insufficient data could have biased results across all groups – but much more likely in the tutorial group.

Besides a shortage of sample points, a slight flaw in the design of our experiments could have caused further bias to the results. We recruited all the participants with a
fixed incentive of 100 dollars pay. Since the participants would receive the same pay regardless of their performance, they have no material incentive to work hard on the assigned exercise. This lack of incentive may have affected the participants in the individual group the most, particularly since they were all working in the same room and were aware of other individuals finishing the exercise and exiting the room. The collaboration and tutorial group participants probably derived some motivation from peer pressure that would offset the bias somewhat. This implies that the actual accuracy difference between the individual and collaboration groups may not be as substantial as our results reflect. Nonetheless, the hypothesis that mutual confirmation improves accuracy is a reasonable one, so we believe that even without the bias, an accuracy difference should still exist.

5.4 Survey

Beyond studying the effects of collaboration, we also hoped to get feedback on the design of the JS prototype. So, at the end of every collaboration and tutorial experiment session, we gave the participants a four-question survey (See Appendix C) about their collaboration experiences. The four questions ask the users to rate the user-friendliness of the prototype, the ease of using the token, the ease of communicating with the group members, and the ease of information sharing compared to face-to-face contact. Each question requested a numerical answer on a scale of 1-7, with 7 being the best rating. Figure 15 shows the averaged results of the survey.
The results show that the participants generally had very positive experiences with the JS prototype. The positive responses to the face-to-face question indicate that virtual collaboration through the JS prototype can work nearly as well as physical collaboration. These results affirm to us the quality of our design.

5.5 Chapter Summary

Using the JS prototype, we performed preliminary experiments to observe student’s behavior collaborative behavior in WebLab. The results show that collaboration slows the students down, but increases their accuracy. Chat logs suggest that this phenomenon could be caused by the repeated process of mutual confirmation between collaborating users. Experiment results also suggest that online tutorials
enhances accuracy even more than student collaboration, although more data points in tutorial mode would be needed to confirm this idea. The data show that collaboration induces some positive effects on students’ work in WebLab, despite possible bias in the results.

The participants’ gave us positive feedback regarding their experiences with the prototype. The positive results about the effects of collaboration and positive feedback about the JS prototype are encouraging to us. They affirm to us that our efforts were useful and that we are proceeding in a good direction.
Chapter 6
Concluding Remarks and Recommendations for Future Work

Collaboration with partners has long been a crucial part of traditional laboratory work. Our experiment results with WebLab indicate that collaboration and tutoring in online laboratories also enhances the quality of individuals' work. Currently, we have a JS prototype that can enable collaboration in WebLab and has produced positive user responses. The JS prototype is, however, just a prototype that marks the beginning of an ongoing effort. Many improvements to the prototype are required before it can replace the single-user WebLab that is currently deployed.

We foresee the following areas of improvement:

- The implementation of a group management system that can handle multiple ad-hoc and permanent groups. Currently, the collaboration server in the JS prototype can handle only one ad-hoc group at a time. This is not sufficient for deployment. We must add group management logic that allows multiple groups to connect simultaneously but collaborate separately. We also need to add to the collaboration server the capability to maintain group information, so that group membership can sustain beyond one collaboration session. To achieve this, the WebLab database should be extended to store information about groups.
• The user interface of the WebLab applet needs an overhaul. When collaborating in the JS prototype, users constantly toggle between five frames on their displays. Having five separate frames in the application clutters the screen and confuses the users. Ideally, each frame should be simplified, and the functionalities of the five frames should be combined into one well-coordinated frame.

• A chat program should be integrated into collaborative WebLab. In our preliminary experiments, we could use third-party chat programs for communication because the experiments were held in a controlled environment. However, we cannot expect all WebLab users to have access to the same chat programs, despite the fact that programs such as AOL IM and ICQ are very widespread. Only when WebLab includes a chat capability can all WebLab users collaborate with each other without any obstacles in compatibility.

• Asynchronous modes of collaboration should be added. The current nearly synchronous mode of collaboration simulates face-to-face collaboration – certainly the most common form of collaboration. Yet, there are asynchronous types of collaboration that can be implemented to expand the potential of a collaborative WebLab. For example, an asynchronous help queue could be very useful. We can create a queue in the database to which a student can submit questions about difficulties he encounters along with the data to which the questions refer. Other students or instructors could view the queue when they log on to the collaboration server, and they can respond to the questions in the queue. The student who submitted the question can then view the responses when he is logged on.
• The application needs to be tested for robustness and scalability. So far, no more than 3 users have connected to the prototype simultaneously. Extensive testing is required to ensure that the application will function as usage of WebLab grows.

We employed the JS prototype as a tool for preliminary testing. We hoped the tests would help us learn about the effects of collaboration on WebLab work and also get usability feedback on our current design. Due to the limited time we had available to prepare and perform the experiments, the quality of the experiments suffered somewhat. Preliminary results were obtained that suggest a direct correlation between collaboration and students’ accuracy and an inverse correlation between collaboration and working speed. We should perform further experiments with significantly more participants to confirm these results (specific recommendation on how to improve these experiments were given in Chapter 5).

An interesting hypothesis to test in future collaboration experiments that we did not test in our preliminary tests is whether there exists a correlation between group size and collaboration effectiveness. The results from such an experiment would help the design of educational WebLab experiments and the design of collaborative WebLab.

The investigation of collaboration in WebLab is an ongoing effort that has only just begun. We have a working prototype and some test results that set the direction for future progress, but a considerable amount of work remains in further development of the system and exploration of its effects on education and learning. Currently, WebLab has been released for use in Singapore and is in the process of further global releases. In the midst of WebLab’s globalization, the development of collaborative WebLab becomes
especially valuable. When deployed, collaborative WebLab can lessen geographic barriers and bring students from different parts of the world together in scientific experimentation and discussion.
## Appendix A

### Communication Protocol Commands

#### A.1 Commands Sent by the Applet to the Collaboration Server

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOIN</td>
<td>Asks the server to join the default group</td>
</tr>
<tr>
<td>DISCONNECT</td>
<td>Requests to be disconnected from the server</td>
</tr>
<tr>
<td>GRAPH</td>
<td>Pushes a set of test results to the server to be broadcasted to other users</td>
</tr>
<tr>
<td>PUSH</td>
<td>Pushes a set of measurement parameters to the server to be broadcasted to other users</td>
</tr>
<tr>
<td>PULL_INPUTS</td>
<td>Pulls the most current set of group inputs from the collaboration server</td>
</tr>
<tr>
<td>PULL_RESULTS</td>
<td>Pulls the most current set of group measurement results from the collaboration server</td>
</tr>
<tr>
<td>PASS_TOKEN</td>
<td>Passes the token to the user whose username is attached</td>
</tr>
</tbody>
</table>
### A.2 Commands Sent by the Collaboration Server to the Applet

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD_USER</td>
<td>Adds the attached username to the Connected Users List</td>
</tr>
<tr>
<td>CONNECTED_LIST</td>
<td>Adds the attached list of usernames to the Connected Users List</td>
</tr>
<tr>
<td>DISCONNECT</td>
<td>Removes the attached username from the Connected Users List</td>
</tr>
<tr>
<td>GRAPH</td>
<td>Graph the attached test results</td>
</tr>
<tr>
<td>PUSH</td>
<td>Update the applet with the attached inputs</td>
</tr>
<tr>
<td>TOKEN HOLDER</td>
<td>Notifies the applet that the current token holder is the user whose username is attached</td>
</tr>
</tbody>
</table>
Appendix B

WebLab Exercise Used in Collaboration Experiments
Appendix C

JS Prototype Survey

WEBLAB / COLLABORATION TOOL SURVEY

Dear student:
Thank you so much for participating in today’s WEBLAB Collaboration Tool experiment. Before you leave today, it would be great if we could take another few minutes of your time to obtain your candid feedback! Your responses are anonymous. Thanks!

User friendliness of Collaboration Tool user interface

1-------------2-------3-- ----- 4----------5--------6---7
Collab.Tool user interface
to figure out
very difficult
Collab.Tool user interface
somewhat easy to figure out
Collab.Tool user interface
very easy to figure out

Collaboration Tool Team “Token”

1-------------2-------3-- ----- 4----------5--------6---7
It was very difficult
To Collaborate with
My team by passing
The team “token”
It was somewhat
easy to collaborate
with my team
by passing the
Team “token”
It was very easy
to collaborate with
my team by passing
the team “token”

Chat / WEBLAB team communication coordination

1-------------2-------3-- ----- 4----------5--------6---7
It was very difficult
to coordinate team
Communication
Using both chat
And WEBLAB Interface
It was somewhat
difficult to coordinate
team communication
using both chat
and WEBLAB interface
It was very easy
to coordinate team
communication using
both chat and
WEBLAB interface
Compared with working face-to-face with team members, how might you rate the ease of communicating numerical information in WEBLAB definition panel using the Collaboration Tool?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>It was much more difficult to communicate numerical information in the definition panel to team members using the Collaboration tool than face-to-face. SIDE!!</td>
<td>It was somewhat more difficult to communicate numerical information in the definition panel to team members using the collaboration tool than face-to-face.</td>
<td>It was just as easy to communicate numerical information in the definition panel to team members when using the collaboration tool as face-to-face. MORE ON OTHER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Please take a moment to suggest improvements in the WEBLAB Collaboration Tool to improve its use in remote team communication:
Describe how you arrived at a protocol for collaboration.

Describe one particular incident in which collaboration helped you. Why was it helpful?
Bibliography


Remote Collaboration in WebLab – an Online Laboratory
by
Victor Chang

Submitted to the
Department of Electrical Engineering and Computer Science

May 16, 1986

In Partial Fulfillment of the Requirements for the Degrees of Bachelor of Science in
Computer Science and Engineering and Master of Engineering in Electrical Engineering
and Computer Science

Thesis Supervisor: Jesus del Alamo