ELVIS iLab: A Flexible Platform for Online Laboratory Experiments in Electrical Engineering

by

Samuel Gikandi

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2006

Author ....
Department of Electrical Engineering and Computer Science
September, 8 2006

Certified by.................... ....
Jesus del Alamo
Professor
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

ELVIS iLab: A Flexible Platform for Online Laboratory Experiments in Electrical Engineering

by

Samuel Gikandi

Submitted to the Department of Electrical Engineering and Computer Science
on September, 8 2006, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis describes a project that is part of the collaboration between MIT and universities in sub-Sahara Africa to exploit the value of iLabs in the developing world. The main goal of this project is to develop software that will exploit the value of the National Instruments Educational Laboratory Virtual Instrumentation Suite (ELVIS) system in Africa by integrating it into the iLabs shared architecture, while taking into consideration the special circumstances surrounding the deployment of iLabs in Africa such as bandwidth limitations, limited access to networked computers and lack of computer skills on the part of students. Integrating ELVIS into iLabs will facilitate the rapid deployment of new online labs to augment the Physics and Electrical engineering curricula in these universities.

iLab development efforts for this project are being done in parallel with developers at the Obafemi Awolowo University (OAU) in Nigeria. One of the main goals of the new system is to fill the gap of laboratory experiences in introductory level electronics and physics classes, which are hardest hit by the lack of equipment due to their typically large enrollment. Our goal is to support the development of electronic circuit building skills by providing an environment where students can easily try different circuit configurations before submitting experiments for execution. We are therefore investigating new iLab client user interface designs that will enable students to create and edit circuit schematics from provided electronic components.

Our ELVIS iLab design will also formalize and simplify the process of creating and administering such labs for instructors, thereby speeding up the deployment of new labs in an environment where software development skills are not at a premium. This will be achieved by recycling many of the components that are currently behind the success of the microelectronics weblab, which have already been adapted before for new iLabs. Besides reusing existing software, the project hopes to make a major contribution towards enhancing students experiences with iLabs through its new interactive client design.

Thesis Supervisor: Jesus del Alamo
Title: Professor

Acknowledgments

I would like to express my gratitude to my Thesis advisor, Prof. Jesus del Alamo, for giving me the opportunity of working on a project that greatlty impacts the way technology education is conducted in universities in Africa. Through this project, I have had the opportunity to interact with highly intelligent and moticvated individuals in the iLab Africa team that are looking to shape the way students learn in developing countries. Getting a chance to contribute in this environment has been very rewarding for me, being from an African country myself.

I would also like to express deep gratitude to the entire iLab team at MIT for providing the atmosphere and tools that I needed in order to bring some of my ideas into fruition. The iLab network was extremely accessible and helpful, especially in getting my software development efforts off the ground.

Lastly, I would like to express my gratitude to our colleagues in Africa, with whom I have worked extensively and developed meaningful relationships. I am also very grateful for the level of hospitality that was accorded to us when we went to the Obafemi Awolowo University (OAU) in Nigeria. The ideas that were generated during that visit were critical towards helping my thesis project move along. It was also a very rewarding experience to realize that there is a growing network of very talented and motivated indivituals in Africa who are committed to improving the quality of higher education in the continent.

Contents

1 Introduction **11**

1.1 Background on iLabs 12

1.2 Background on iLabs in Africa 12

1.3 The NI-ELVIS iLab 13

1.4 Overview of this Thesis 14

2 ELVIS and iLabs: Bringing the two together **17**

2.1 ELVIS Platform 17

2.1.1 NI ELVIS workstation 19

2.1.2 NI ELVIS and LabVIEW 19

2.1.3 Why the ELVIS system is suitable for deploying iLabs in Africa 21

2.2 iLabs Shared Architecture 23

2.2.1 The iLab Framework 24

2.2.2 The batched experiment Architecture 26

2.3 Vision for an online ELVIS 28

2.3.1 Make the instructors job easy 28

2.3.2 Minimize new software development requirements 28

2.3.3 Make the students experience as real as possible 29

2.3.4 Costs versus number of experiments 29

2.3.5 Adding value to iLabs 29

3 ELVIS Weblab: Initial Approaches **31**

3.1 First versions of ELVIS iLab software 31

3.1.1 MIT Approach: ELVIS weblab v.1 31

3.1.2 OAU Approach: The OpLab 41

3.2 Motivation for the new software architecture 45

3.3 Specifications of a new design 46

4 ELVIS Weblab v2: A unified Approach **49**

4.1 Architecture Overview 49

4.2 Design Process 50

4.2.1 Describing the Experiment Domain 50

4.2.2 XML Document Specification 52

4.2.3 Lab Server 54

4.2.4 Lab Client 60

4.3 Results 62

5 Conclusion and Future Work **65**

5.1 Future iLab-Africa software development efforts 65

A XML Specification Documents **69**

A.1 ELVIS Weblab v1 69

A.1.1 LabConfiguration.xml 69

A.1.2 ExperimentSpecification.xml 70

A.1.3 ExperimentResult.xml 71

A.1.4 ExperimentSpecification.xml 74

A.1.5 ExperimentResult.xml 76

B Report of visit to OAU **77**

List of Figures

2-1 This diagram shows the three components that make up the ELVIS platform. 18

2-2 This is a diagram of the ELVIS workstation with a simple circuit constructed on its prototyping board. The knobs on the front can be used to control the workstation manually. 19

2-3 The ELVIS instrument launcher gives users access to LabVIEW built virtual instruments that can control the ELVIS workstation. The source code for these instruments is provided and can be added to LabVIEW virtual instruments. 20

2-4 This figure shows an overview of the three-tiered iLab system. The Service Broker handles all administrative tasks, thus freeing the Lab Server (and its developers) from having to implement custom administrative solutions for each different weblab. The Service Broker architecture also simplifies the sharing of iLabs between universities, by alleviating the lab-side (host) university from administering guest users. The host university can grant access to the student-side (guest) universitys Service Broker, and the guest university can then administrate its own users. 25

3-1 This diagram shows the modifications made in order to incorporate the ELVIS workstation into the Microelectronics weblab sofwtare architectiure. 33

3-2 This diagram shows the modules used to implement the Microelectronics weblab server and how they interact with each other. 33
3-3 This diagram shows how the experiment execution engine communicates with the ELVIS work station using virtual instruments. The numbered arrows indicate the flow of method calls and parameters from the experiment execution engine to the dll that conducts the experiment and back. 37
3-4 A comparison of the front end of the Microelectronics weblab client and the ELVIS weblab client. The striking similarity demonstrates how most of the concepts used to implement the Microelectronics weblab were recycled in the first version of the ELVIS weblab. 39
3-5 A comparison of the dialogs used by the user to specify parameters for experiment execution for the Microelectronics weblab client and the ELVIS weblab client. 40
3-6 This page of the OAU opLab client presents the user with a choice of operational amplifier configurations. 42
3-7 This page of the OAU opLab client presents the user with the components required to create an inverting opAmp configuration. The user is then required to figure out which components to connect together before submitting the experiment to the lab server for execution. . . . 43
3-8 This page of the OAU opLab client displaye the results of executing an experiment to the user 44

4-1 This figure shows the classification of circuit components that the user can interact with and their features. This model motivates the design of other lab server and client components. 52
4-2 This figure shows the database structure used to store information about experiment setups for ELVIS weblab v2. Note: In the diagram, PK refers to Primary Key while FK refers to Foreign Key. 58

Chapter 1

Introduction

The need for hands on experience in electrical engineering and physics classes at the university level cannot be overstated. Students have been shown to develop a much better understanding of complex technical concepts when teaching is complemented by laboratory experience. The value of interacting with live devices through experiments to complement theoretical models presented during lectures cannot be overstated [7].

However, the administration of laboratory experiments to students requires a large investment in time, money and personnel. This is especially evident in developing countries, where the inadequacy of laboratory space, equipment and personnel coupled with the over-enrollment of students in technical courses aggravate the situation. Coping with these demands can lead to a dearth of experiments in technical classes, which impedes on the learning experience for students enrolled in these classes.

In order to address this need, three Universities in Africa are now turning towards online laboratories, particularly those implemented using the MIT iLabs architecture, to improve their science and engineering curriculum. The MIT iLabs architecture facilitates the deployment, management and sharing of online laboratories that interface with real hardware devices [5, 3]. This interest has led to a rich collaboration between MIT and the universities in Africa, dubbed the iLab Africa Project [17].

This thesis outlines one of the projects that have been started under the auspices of this initiative. The project involves the development of a software framework within which to create and deploy new iLabs based on the National Instruments Educational Laboratory Virtual Instrumentation Suite [13].

1.1 Background on iLabs

The iLabs project was conceived in 1998 by Prof. Jesus del Alamo to provide students with hands on laboratory experience in classes where equipment may be expensive and difficult to set up. The project also facilitates easy sharing of labs across institutions and simplifies the logistics involved in administering labs to large groups of students. Since its inception, the concept has been adopted in physics, electrical engineering, chemical engineering and mechanical engineering classes, and has been successfully deployed and tested by students and faculty at MIT and across the world [3, 6, 22].

The concept of iLabs has been very well received by both instructors and students. Instructors enjoy the benefits of a greatly simplified logistics model when administering laboratory experiments and accessing laboratories that have been built using devices located at other universities. Students on the other hand value the ability to access sophisticated and expensive devices at any time of the day, which gives them hands on experiences in classes where the organization of such labs may otherwise be too complex or expensive [11].

1.2 Background on iLabs in Africa

The idea of introducing iLabs to Africa was proposed in 2003 by Prof del Alamo. This was followed by a feasibility study to determine the potential of iLabs in enriching the educational experience for university students and the problems that implementation of iLabs in Africa would potentially face. The study was conducted at three leading universities: Nigerias Obafemi Awolowo University (OAU), Tanzanias University of Dar es Salaam (UDSM) and Ugandas Makerere University [3].

The study showed that the iLabs concept is very relevant and could be beneficial to African universities for several reasons. First, the course material that iLabs complement is widely taught to many students at these universities. These universities also have world class faculty and enthusiastic students who could help adopt and propagate the iLabs concept. The idea of iLabs could also help alleviate the lack of adequate physical laboratories and equipment, which deprives many African students of hands on experience in their engineering and physics curriculum.

The study also noted that the current situation in Africa requires the development of solutions unlike those used at MIT in order to make the deployment of ilabs effective. Some of the factors which led to this conclusion include limited access to networked computers, limited student exposure to computers, severe bandwidth limitations, power outages and fragile local networks at these institutions.

As a result of this study, the iLabs Africa project was officially launched in the summer of 2005. The goals of the project include the deployment of current MIT iLabs throughout the curriculum in African universities, the development of new labs within the iLabs architecture, creating opportunities for internships for MIT and African students and staff and the creation of a scalable iLabs research network in Africa.

Since this study was conducted, a lot of progress has been made towards realizing the goals of the iLab Africa project. iLab Africa Teams have been created at MIT and at all three universities in Africa in order to facilitate the development, deployment and sharing of iLabs at the African universities. The project has also seen several exchanges between student and faculty at the collaborating institutions to facilitate the exchange of expertise and ideas. More recently, collaboration between developers at MIT and at OAU has led to the development and deployment of new iLabs. These are the software efforts that form the basis of this MEng thesis.

1.3 The NI-ELVIS iLab

The NI ELVIS iLab project was born when students and faculty from OAU, who had come to MIT over the summer of 2005, were given a donation of an ELVIS board [13] and LabVIEW [14] software to take back to Nigeria with them. Since then, the iLab

Africa team at MIT has been working closely with them to develop new content and iLabs based on the ELVIS platform.

The goal of the ELVIS iLab project is to use the ELVIS platform as a launching pad for guiding future iLab software development efforts in Africa, while enhancing the collaboration between MIT and African universities and adding value to the current iLabs architecture. Eventually, the experience accrued from these exchanges should form a basis for enabling OAU to become a hub for the development and deployment of iLabs in Africa.

A major component of successful completion of the project was a visit to OAU by MIT students in January 2006, where two of us spent three weeks working with the iLab team to determine the best approach for deploying online laboratories based on the ELVIS platform and familiarizing ourselves with the resources available and problems faced while developing labs to be incorporated into the iLabs software architecture.

As it turned out, this visit was crucial in advancing the software development efforts of the iLab Africa project, especially since both groups (MIT and OAU) had been working independently on separate solutions for incorporating ELVIS into iLabs. The exchange of ideas during the January visit has led to a unified software development approach, which takes advantage of the most promising features of each of the two approaches and heralds a new, enhanced architecture that should speed up the deployment iLabs based on ELVIS in future. This unified approach is discussed in detail in Chapter 4 of this document.

1.4 Overview of this Thesis

Chapter 2 of this thesis will discuss the ELVIS system in greater detail and explain some of the factors that make it especially suited for developing and deploying iLabs in Africa. This chapter will also discuss the iLabs shared architecture in detail to provide a background for the software architecture discussed in chapter 4 of this document, as well as outline characteristics that make this architecture suitable for development of online laboratories in developing countries. This chapter will conclude with a discussion of out vision for an online ELVIS platform.

Chapter 3 will discuss the different software approaches that were initially implemented by the MIT and the OAU teams to incorporate ELVIS based experiments into the iLabs Shared Architecture. This chapter will focus on the strengths and weaknesses of these approaches, thereby motivating the implementation of a unified software architecture that captures the strengths of both approaches and alleviates their weaknesses.

Chapter 4 will contain a detailed discussion the unified software architecture motivated in chapter 3. This chapter will also discuss the results of this software implementation and its effectiveness in meeting the requirements specified in chapter 3.

Chapter 5 will wrap up the thesis with a discussion of proposed future work to enhance the success of software development efforts for iLabs in Africa.

Chapter 2

ELVIS and iLabs: Bringing the two together

This chapter introduces the tools used to facilitate the software development efforts for the iLab Africa project. In particular, this section discusses the features of the National Instruments Educational Laboratory Virtual Instrumentation Suite (NI ELVIS) platform, LabVIEW software and the iLabs shared architecture that make them suitable for developing and deploying iLabs in developing countries.

2.1 ELVIS Platform

The National Instruments Educational Laboratory Virtual Instrumentation Suite is a LabVIEW-based design and prototype environment for university science and engineering laboratories [4]. NI ELVIS consists of LabVIEW-based virtual instrument suite, a multifunction data acquisition (DAQ) device and bench-top workstation with a prototype board. A diagram of the ELVIS system is shown in figure 2-1.

NI ELVIS functions as a three-part system. The ELVIS workstation interfaces with National Instruments LabVIEW software and an NI data acquisition (DAQ) device to perform measurements and transmit signals.

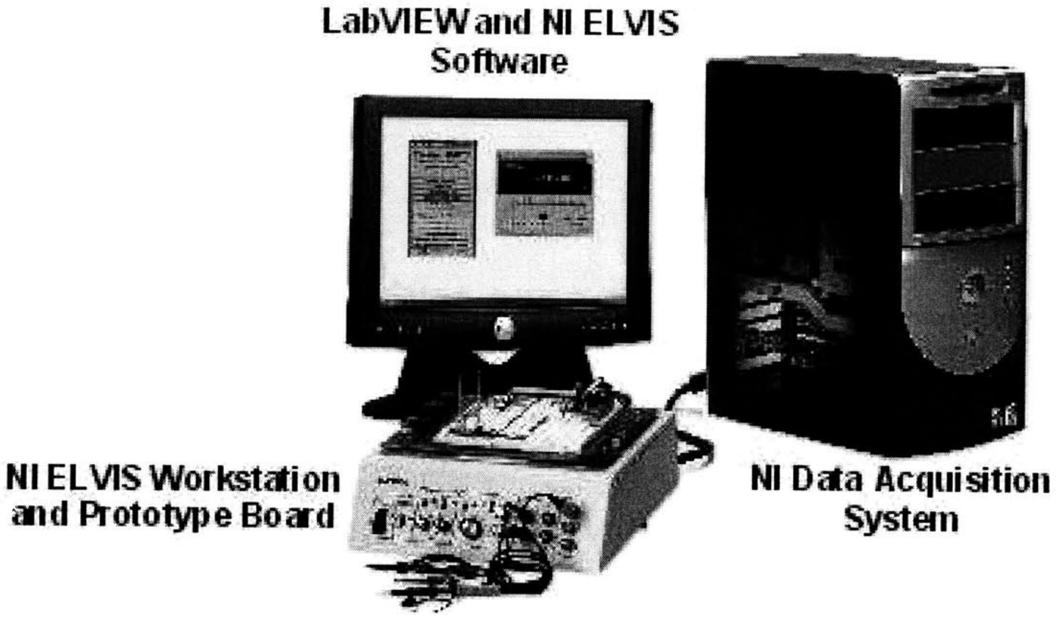

LabVIEW and NI ELVIS Software
NI ELVIS Workstation and Prototype Board
NI Data Acquisition System

Figure 2-1: This diagram shows the three components that make up the ELVIS platform.

2.1.1 NI ELVIS workstation

The ELVIS workstation can be manipulated by users either from the built-in control panel located ion the front of the workstation, or programmatically through the included NI ELVIS software suit (see figure 2-2 below).

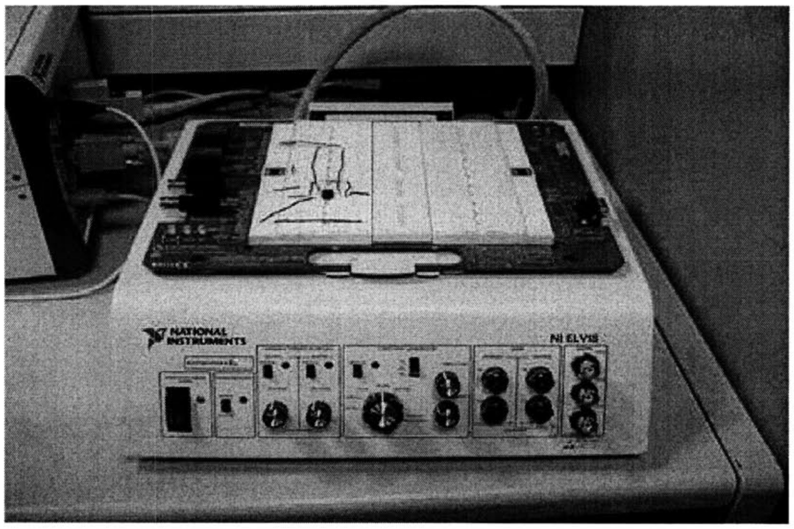


Figure 2-2: This is a diagram of the ELVIS workstation with a simple circuit constructed on its prototyping board. The knobs on the front can be used to control the workstation manually.

The workstation also comes with a removable prototype board with over 2800 tie points. This board can be used to build electrical circuits and connect them to the programmatically controlled instruments that come with ELVIS. In addition to variable power supplies that are included in the instrument suit, the board also offers built in 15 Volts and +5 Volts power supplies, which can be used to build a wide array of circuits.

2.1.2 NI ELVIS and LabVIEW

LabVIEW [14] is a graphical programming environment for test, measurement, and automation applications. The main benefit of LabVIEW as far as this project is concerned is its extensive support for accessing instrumentation hardware. Drivers and abstraction layers for many different types of instruments are available and are presented as graphical nodes. This abstraction layer presents as standard software interface for communicating with a wide range of hardware devices. The provided driver interfaces also saves program development time. Because of its flexibility, modular nature, and ease of programming, LabVIEW has become a language of choice for laboratory use in top universities worldwide [16]. In fact, one of the future goals of the iLabs project is to develop software that will integrate LabVIEW into the existing iLabs architecture [1].

The ELVIS platform comes with LabVIEW based software virtual instruments, which can be modified using the LabVIEW application programming interface to create custom instruments and build elaborate lab exercises to suit the needs of instructors (See figure 2-3). The source code for all the included ELVIS virtual instruments is available to developers for possible modification and extension.

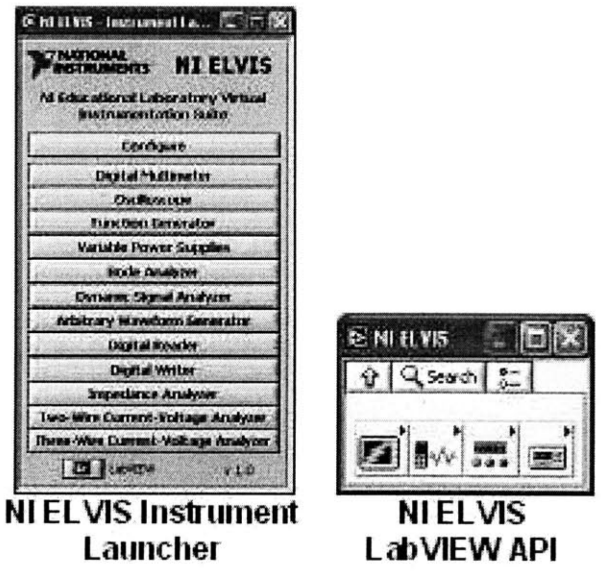


Figure 2-3: The ELVIS instrument launcher gives users access to LabVIEW built virtual instruments that can control the ELVIS workstation. The source code for these instruments is provided and can be added to LabVIEW virtual instruments.

The software instruments included with the ELVIS platform can be categorized into two:

- Signal sources: Function Generator, Variable Power Supplies, Arbitrary Waveform Generator, and Digital Writer

- Signal analyzers: Oscilloscope, Dynamic Signal Analyzer, Digital Multimeter, Bode analyzer, and Current-Voltage Analyzers

As an example of how these instruments integrate with the ELVIS hardware, a waveform can be generated using two different methods: using the built-in full-featured function generator that can be controlled using knobs on the control panel or programmatically with a specially designed LabVIEW software suite included with NI ELVIS.

Because of the workstation prototype boards design, the user has the added capability of connecting signals between various instruments and DAQ system inputs. For example, the output of the function generator can be connected to a specific input channel of the DAQ board, ultimately displaying it on a desired channel of the oscilloscope included with the instrument suite [4].

2.1.3 Why the ELVIS system is suitable for deploying iLabs in Africa

There are several attributes of the ELVIS system that make it an attractive platform for developing iLabs for this project:

Basic to complex experiment support

During our visit to Nigeria, we observed the greatest need for laboratory equipment in the electrical engineering and physics departments at the introductory level of the curriculum. Introductory level classes are extremely over subscribed at the university, putting tremendous pressure on scarce laboratory space and equipment. It is not uncommon for single workstations to be shared amongst ten or more students. Access to lab space at this stage is also time-limited, with large groups of students taking turns to attend the lab for one to two hour shift (see Appendix b for a detailed report on the visit to Nigeria).

From our discussion with faculty at the university, we also noted that access to advanced electrical engineering equipment could lead to a marked improvement in the variety and effectiveness of courses offered to students taking advanced classes. A good example of this is the positive effect that access to the Microelectronics weblab [20], which is implemented using the iLabs Shared Architecture, has had on the corresponding course at the university.

As its name suggests, the ELVIS platform was primarily designed to support experiments conducted within an educational context. The three components of the platform make it extremely versatile, enabling it to support experiments that range from the most basic to those conducted in project based classes [19] There is therefore immense potential in a software architecture that complements the versatility of ELVIS with the ability to share and administer laboratories amongst large groups of students provided by the iLabs Shared Architecture.

LabVIEW integration

The LabVIEW programming environment is widely used in educational and industrial environments [14]. As a result, there is a tremendous amount of documentation and support available, as well as a host of educational experiments for university level courses implemented using LabVIEW. The success of LabVIEW can be attributed to the fact that it is very versatile and intuitive to use, even for users with limited programming experience. It is also a platform that is well understood by faculty members in Africa, as we noted during our visit to Nigeria in January.

It is therefore of great importance that the ELVIS platform comes complete with a suite of LabVIEW virtual instruments, as this provides us with access to most of the ELVIS functionality within an environment that instructors are already familiar with.

Since one of LabVIEWs strongest selling points are its support for accessing instrumentation hardware, a software framework that communicates with ELVIS through LabVIEW could also form a basis for developing new labs that are built to interface other LabVIEW-ready hardware equipment with iLabs. This framework would open iLabs to a wide range of equipment that is used through out the Engineering curriculum, increasing the range and effectiveness of iLabs in Africa.

Easy to use Interface

One of the characteristics of good software architecture for creating online laboratories should be the ease with which new labs can be deployed in an environment where software development skills are limited. This is the case at OAU, where the high student to faculty ratio means that instructors frequently take on administrative functions and have little time to conduct research, let alone develop software. This dearth of skills is compounded by the fact that research funding for students is not easily acquired.

The ease with which ELVIS integrates with LabVIEW makes it a good choice in this respect. The ELVIS platform includes LabVIEW based virtual instruments, which can be readily accessed and used to create and deploy experiments. This easy integration with LabVIEW also gives ELVIS the computer connectivity required to for interfacing with the iLabs shared architecture.

Wide Use and support

As mentioned in its manual, the ELVIS system was specifically designed for use within an educational context. As a result, there are already a large number of universities that are developing experiments based on ELVIS to supplement their engineering and science curriculum [15]. Providing a way of connecting ELVIS to iLabs could therefore enhance the sharing of a wide range of experiments with immense pedagogical value.

2.2 iLabs Shared Architecture

The iLab project was developed as part of the iCampus initiative [10] at MIT to enhance the learning experience of students in science and engineering by providing more hands-on experience. The project was developed to alleviate some of the problems that are common with conventional laboratories.

Some of the problems associated with conventional laboratories include high cost and complex logistics. Also, if expensive equipment is used, administering conventional laboratories requires the presence of trained personnel. Even this does not guarantee the safety of the equipment, which often results in students being denied access to sensitive equipment that is costly to replace if damaged during an experiment. Another problem arises when the laboratory equipment being used is not widely accessible, with time-sharing schemes being required to provide all students access to the equipment. Conventional labs may also not scale well; the above problems increasing dramatically with the number of students [2].

Online labs share the advantages of conventional labs while overcoming some of their limitations. They provide hands on experience with real devices, which can be accessed from any location and at any time over the internet. Besides being convenient for students, who no longer have to go to the lab in order to conduct experiments on these devices, online labs are also very flexible and easy to manage from the point of view of the instructors. Online labs also give students access to experiment data, allowing them to analyze the data using different software packages [22].

The motivations for the development of online laboratories are especially pronounced in developing countries. The problem of scarce laboratory equipment and space is aggravated by a large demand due to over-enrollment in many engineering disciplines. The high student to faculty ratio in universities in these countries also makes lab administration extremely difficult to manage. As a result of these factors, many students do not get a chance to interact with real hardware devices often in their curriculum, a component that is extremely useful to reinforce concepts presented in class.

2.2.1 The iLab Framework

The iLab framework refers to a software architecture that was created at MIT in order to facilitate the creation, management and distribution of laboratory exercises designed for Engineering and science courses. The software framework provides the functionality necessary to give geographically distributed users access to real electrical devices. Consequently, the iLab framework provides a set of generic services relating to user authentication and authorization, group management, experiment specification and result storage, as well as lab access scheduling. This architecture is demonstrated in Figure 2-4[21].

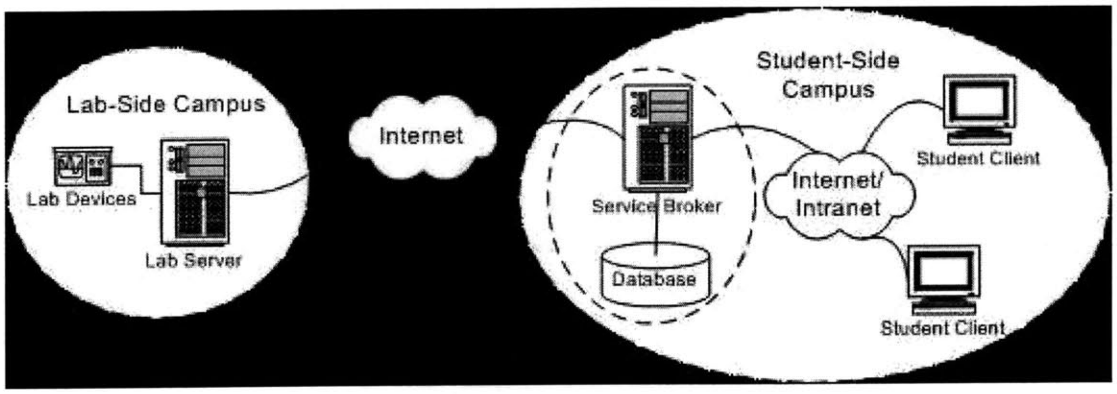


Figure 2-4: This figure shows an overview of the three-tiered iLab system. The Service Broker handles all administrative tasks, thus freeing the Lab Server (and its developers) from having to implement custom administrative solutions for each different weblab. The Service Broker architecture also simplifies the sharing of iLabs between universities, by alleviating the lab-side (host) university from administering guest users. The host university can grant access to the student-side (guest) universitys Service Broker, and the guest university can then administrate its own users.

The long term vision of the iLab project is for the educational content of online labs to be broadly shared around the globe, enhancing science and engineering education by multiplying the lab experiences students will ultimately have access to [2].

The iLab framework defines and supports three broad categories of experiments [5]:

- **Batched experiments:** In this experiment type, the user submits a set of specifications informing the lab hardware what parameters to use to conduct the entire experiment. The lab hardware then conducts the requested experiment and sends the resulting values back to the user. In this experiment type, therefore, the user does not need to maintain connection with the lab server over the duration of the experiment.

- **Interactive experiments:** In this case, the user can dynamically access and change various parameters over the duration of the experiment. The user therefore needs to maintain an internet connection with the lab server as the experiment is being conducted.

- **Sensor experiments:** In this experiment type, users monitor or analyze real time data without influencing the phenomenon being measured.

Of these three experiment types, only batched experiments have been fully tested and deployed currently. Nonetheless, this is the experiment type that lends itself particularly well to deployment in developing countries, where conditions such as limited bandwidth, limited computer access and network failures are prevalent.

2.2.2 The batched experiment Architecture

The current batched architecture (dubbed the iLabs shared architecture) is based on a 3-tier system [5, 3]. The three components are:

- A **client** that displays a list of existing experiment setups and allows the user to specify parameters that will determine what values will be used when the experiment is conducted. These experiment specifications are sent via an XML document to the service broker, which forms the second tier in this architecture.

- A **service broker**, which manages communication between the client and the lab server.

- A **lab server** that manages communication with the actual hardware and conducts experiments requested by the user. The lab server sends the results of the experiments to the service broker, which notifies the client that the experiment has been completed.

The service broker in this architecture consists of completely lab independent and generic code. It handles the authentication of students wishing to gain access to different labs by assigning them to groups. These groups, which usually correspond to courses offered in participating universities, are then passed on to the lab server, which determines whether to grant them access to the hardware.

The lab server and the client on the other hand need to understand a common protocol for describing the nature of the experiments to be conducted. This protocol is completely abstracted away from the service broker, which passes on messages as-is from the lab server to the client and vice versa.

The lab server and the service broker are currently implemented as .NET web services [18]. They both export methods which can be called through the SOAP [23] standard, which provides a platform independent standard for exchanging information in a decentralized and distributed environment. The lab client is usually implemented using technologies that can be accessed from web browsers. Currently, the client is implemented as a Java applet, with a back end that supports communication with the service broker via the SOAP standard.

In order for iLab-enabled online laboratories (referred to as weblabs from here on) to be truly distributed, the different layers must adhere to the set of standardized operations stipulated in the iLab API. These operations are implemented over web services and the SOAP protocol for message exchange. The iLab shared architecture defines two sets of web service methods composing the iLab API: service calls from Lab Client to Service Broker [24], and service calls from Service Broker to Lab Server [9]. In fact, the bulk of the iLab API consists of pass-through methods, whose function is simply for the Lab Client to call a corresponding method from the Lab Server API.

The most important methods are listed below:

- *GetLabStatus*: checks on the status of the Lab Server.
- *GetEffectiveQueueLength: checks on the effective queue length of the Lab Server.*
- *GetLabInfo: gets general information about a Lab Server.*
- *GetLabConfiguration: gets the lab configuration of a Lab Server.*
- *Validate: checks whether an experiment specification would be accepted if submitted for execution.*
- *Submit: submits an experiment specification to the Lab Server for execution.*
- *Cancel: cancels a previously submitted experiment.*
- *GetExperimentStatus: checks on the status of a previously submitted experiment.*

- *RetrieveResult: retrieves the results from a previously submitted experiment.*

The iLab architecture provides a lot of the functionality that is required to set up stable and deployable weblabs with minimal software development efforts. Institutions hoping to put up new iLabs can have access to the service brokers functionality without having to write any software. The lab server and client software is also easy to customize for new experiments as we will see in the next two chapters.

The iLab architecture also renders itself particularly well to sharing of online laboratories amongst different institution, a feature that would be invaluable for students in developing countries hoping to access equipment that is not available at their institution.

2.3 Vision for an online ELVIS

The goal of this project is to determine how best to develop solutions that will enhance the experiences of both users and instructors with an ELVIS weblab. The clear benefit of putting ELVIS online is greater access to all its instrumentation capabilities, while offering instructors in developing countries an alternative to conventional laboratories that is stable, easy to set up and easy to manage. The design of the system that is proposed in this project is motivated by the following goals:

2.3.1 Make the instructors job easy

We want to make the instructors job as easy as possible. It is easy to create new experiments on the ELVIS platform and it should therefore be easy to deploy these experiments on iLabs

2.3.2 Minimize new software development requirements

We want to provide a very generic software framework that will take care of most of the steps involved in creating and managing ELVIS weblabs, thereby reducing the magnitude of software development work required to create new experiments.

This will involve encapsulating the administrative aspects of the experiment into a pluggable module and capturing the nature of experiments built on ELVIS using a generic software model. This software model should capture all the aspects involved in creating, managing and displaying an experiment on ELVIS.

2.3.3 Make the students experience as real as possible

One of the main observations made about users of online laboratories is their skepticism about whether they are communicating with actual hardware or not. In fact, it has been observed that the more students believe that they are communicating with actual hardware as opposed to simulations, the more value they derive from the online experiments [21]. As a result, a substantial portion of whether the project will be successful will depend on providing a client that mirrors the hardware as best as possible, providing the kind of flexibility that students expect with actual hardware.

2.3.4 Costs versus number of experiments

The ELVIS system itself costs about 2000 US Dollars, which can be prohibitive in developing countries. In fact, the initial advances made in this project are largely due to donations of hardware and software made by National Instruments. In order to extract maximum value from a single ELVIS board therefore, it is desirable to be able to create more than one experiment at a time on the board. This is largely an exercise in circuit design, as well as choosing the right hardware that will multiplex some of the channels available to connect ELVIS instruments to the actual hardware

2.3.5 Adding value to iLabs

Besides being deployed and used by students around the world, another main goal of the new ELVIS weblab is to add value to the existing iLabs software architecture. This can be achieved by providing components that improve on the existing capabilities of already implemented weblabs. This motivation led to a lot of new ideas emerging during the implementation of the ELVIS weblab.

Chapter 3

ELVIS Weblab: Initial Approaches

When this project was conceived, the iLab Africa teams at OAU and MIT adopted different approaches to incorporate the ELVIS workstation into the iLabs shared architecture. This chapter discusses these approaches in detail and outlines the strengths and weakness of each approach. The chapter concludes by motivating a unified approach software development approach to help the iLab Africa project grow to the next level.

3.1 First versions of ELVIS iLab software

Over the fall semester of 2005, we at MIT worked in parallel with the iLab team at OAU to develop lab server and client software that would help connect the ELVIS platform to the iLabs shared architecture. This section will outline the main ideas followed in the development of these two approaches and then motivate the need for a unified approach that leverages the lessons learned during the development of these initial versions.

3.1.1 MIT Approach: ELVIS weblab v.1

The first version of the ELVIS software (dubbed ELVIS weblab v1) that we developed here at MIT recycled most of the software that already existed for the Microelectronics weblab [6, 20]. The MIT Microelectronics Weblab is an online semiconductor characterization laboratory that has been successfully deployed and tested in universities around the world.

By working with deployed components, our approach enabled us to create a stable release that contained a lot of the functionality that had already been worked into an existing weblab and thoroughly tested. In order to develop a weblab for ELVIS, we made changes that enabled the existing software to conform to a new software model for representing experiments built on ELVIS. The key difference is that while the user interacts with a single physical *device* on the client side in the Microelectronics weblab (such as a diode or a transistor), the user in the ELVIS weblab interacts with an *experiment setup*, which represents a complete circuit that is constructed on the ELVIS prototyping board for that particular experiment.

As a result, the functionality of ELVIS weblab v.1 closely mirrors that of the Microelectronics weblab, with the main difference being the terminology used to describe experiments. Despite this difference, there is nonetheless an almost a direct mapping between how *devices* are represented in the Microelectronics weblab to how *experiment setups* are represented in the ELVIS weblab. Modifying the Microelectronics weblab software to work with ELVIS was therefore reduced to a rather straightforward task that mainly involved changing the terminology used in the existing software. Figure 3-1 outlines this process graphically.

Lab server

The Microelectronics weblab lab server is implemented as a three-tier .NET web application as shown in 3-2 [6, 8]. The first tier consists of an SQL database, which stores all the information that supports the servers functionality. The data in this database is manipulated by a set of visual basic modules (Data management interfaces), which export functions to be used by the .ASP user interface layer. The server communicates with the experiment hardware through a stand-alone Experiment Execution Engine and exports its functionality via web services using the SOAP standard.

The main components of the Microelectronics weblab lab server that were modified

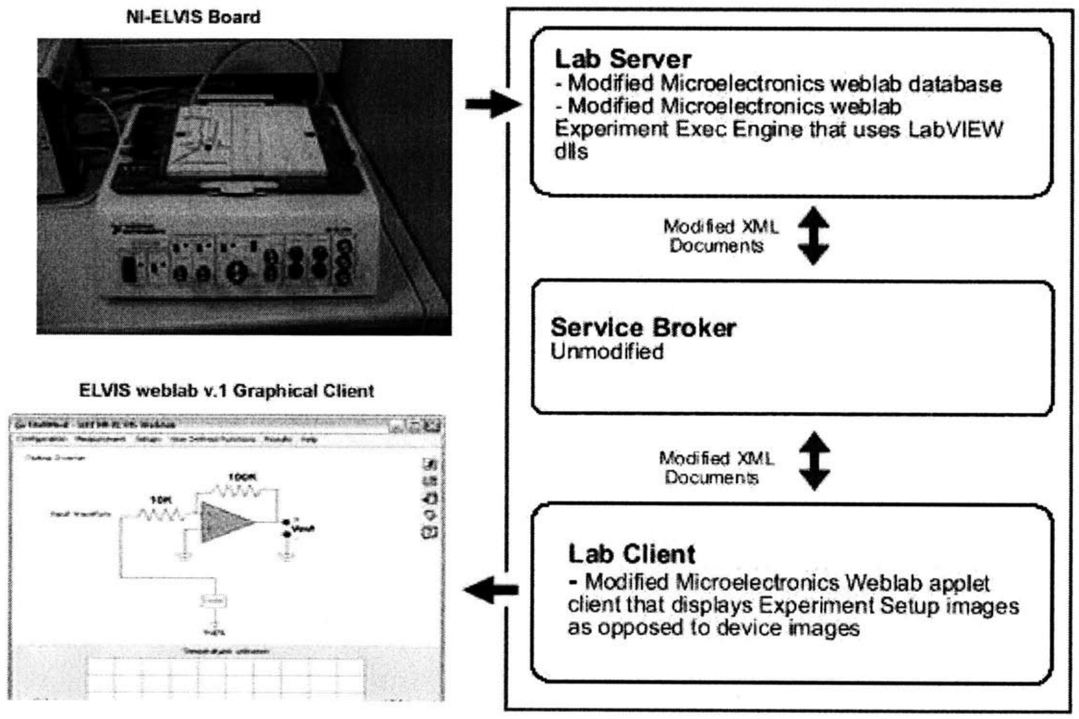

NI-ELVIS Board
ELVIS weblab v.1 Graphical Client
Lab Server
- Modified Microelectronics weblab database
- Modified Microelectronics weblab Experiment Exec Engine that uses LabVIEW dlls
Modified XML Documents
Service Broker
Unmodified
Modified XML Documents
Lab Client
- Modified Microelectronics Weblab applet client that displays Experiment Setup images as opposed to device images

Figure 3-1: This diagram shows the modifications made in order to incorporate the ELVIS workstation into the Microelectronics weblab sofwtare architectiure.

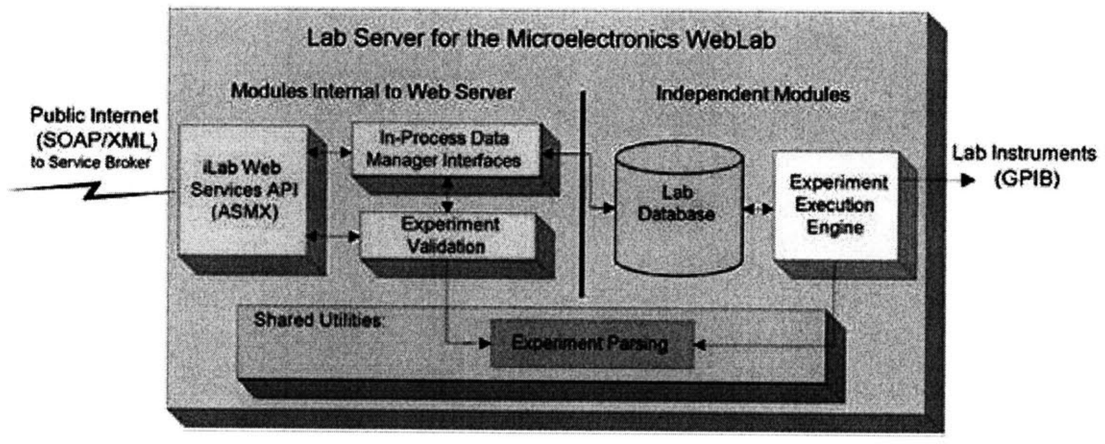

Lab Server for the Microelectronics WebLab
Modules Internal to Web Server
Independent Modules
Public Internet (SOAP/XML) to Service Broker
iLab Web Services API (ASMX)
In-Process Data Manager Interfaces
Experiment Validation
Lab Database
Experiment Execution Engine
Lab Instruments (GPIB)
Shared Utilities:
Experiment Parsing

Figure 3-2: This diagram shows the modules used to implement the Microelectronics weblab server and how they interact with each other.

in order to implement the ELVIS weblab lab server are:

- The Lab Configuration XML document: This document is created by the lab server to inform clients about the experiment setups that are available for execution. Whereas the layout of the document was maintained, the terminology used was changed to convey information about Experiment Setups as opposed to Devices.

- The device specific tables in the lab server database: These are the tables that store the information necessary to generate the Lab Configuration XML document described above. The names of these tables and corresponding columns were changed in order to conform to the new terminology.

- In order to work with the new database structure, the Visual Basic back end and the ASP front end of the lab servers device specific sections also needed to be modified.

- The Experiment Execution Engine: This module was modified substantially to enable it communicate with ELVIS hardware. These modifications are described in detail below.

These changes did not impact the functionality of the lab server in any major way. The process of creating and managing new experiments was still strikingly similar to that followed in the Microelectronics weblab, as was the rest of the administrative functionality.

Experiment Execution Engine

The only component of the lab server whose functionality was greatly modified was the Experiment Execution Engine. This is the stand alone module that interacts with the hardware and runs experiments that are queued in the lab servers database. This module is radically different from its corresponding counterpart in the Microelectronics weblab because it communicates with an entirely different piece of hardware (the ELVIS workstation as opposed to a parameter analyzer).

For the first version of the ELVIS weblab, the experiment execution engine communicates with a simple Operational Amplifier circuit that is set up in the inverting configuration. The approach used to communicate with the actual hardware required the following steps to be completed in order to set up the lab exercise:

1. The circuit representing the lab exercise was constructed on the ELVIS prototyping board. For this first implementation, the lab server was limited to conducting only one experiment at a time.

2. A LabVIEW Virtual Instrument (VIs) was then constructed to handle communication with the ELVIS board for this experiment. The virtual instrument is responsible for:

   - Initializing communication with the ELVIS board. This was achieved by interfacing our custom virtual instrument with the virtual instrument responsible for initializing communication with ELVIS board in the source code for ELVIS.

   - Driving the circuit built in the first step with a function whose parameters (waveform type, frequency, amplitude and offset) are to be specified by the user. This was achieved by interfacing our virtual instrument with the Function Generator virtual instrument that came bundled with the ELVIS instrumentation suit.

   - Reading the waveform values that represent both the input and output values from the ELVIS workstation at predetermined ports. This task was achieved by a custom LabVIEW data acquisition virtual instrument (the DAQ assistant) that reads data from a specified port at a given sampling rate.

   - Closing the connection with ELVIS and releasing all the resources necessary to communicate with it. This task is again delegated to a virtual instrument that came with the source code for ELVIS.

3. The virtual instrument built in step 2 was then exported as a C++ dynamic link

library (dll). The functionality required to build dlls from virtual instruments is provided within the LabVIEW development environment [12]. It allows each virtual instrument to be represented in software as a single function that can take in a set of specified arguments. In our case, the VI was built into a dll that takes in inputs to specify the type, frequency, amplitude and offset of the waveform to use to drive to the circuit and returns as the output an array of values that represent the resulting input and output waveforms.

4. The final step involved creating visual basic wrapper classes around the C++ dlls [12]in order to enable them interoperate correctly with the code used in the experiment execution engine. This allows the experiment execution engine to handle communication with the ELVIS hardware using regular visual basic calls.

Figure 3-3 shows the resulting cascade of calls that occurs in order for the experiment execution engine to communicate with ELVIS.

Despite having these modifications, the experiment execution loop in the experiment execution engine is identical to that used in the Microelectronics weblab. The key steps followed are:

1. Checking the database for any queued experiments. If these exist, the record associated with that experiment is fetched.

2. The Experiment Specification XML (See appendix A Experiment Specification XML Document) document stored with that record is parsed to extract the ID of the experiment to conduct and the parameters to use when conducting the experiment.

3. The validation engine module is called. This module checks to see whether the inputs passed in are legal and within the specified limits. If this is not the case, an error message is constructed and sent back to the user.

4. If the parameters specify a valid experiment, this experiment is conducted by

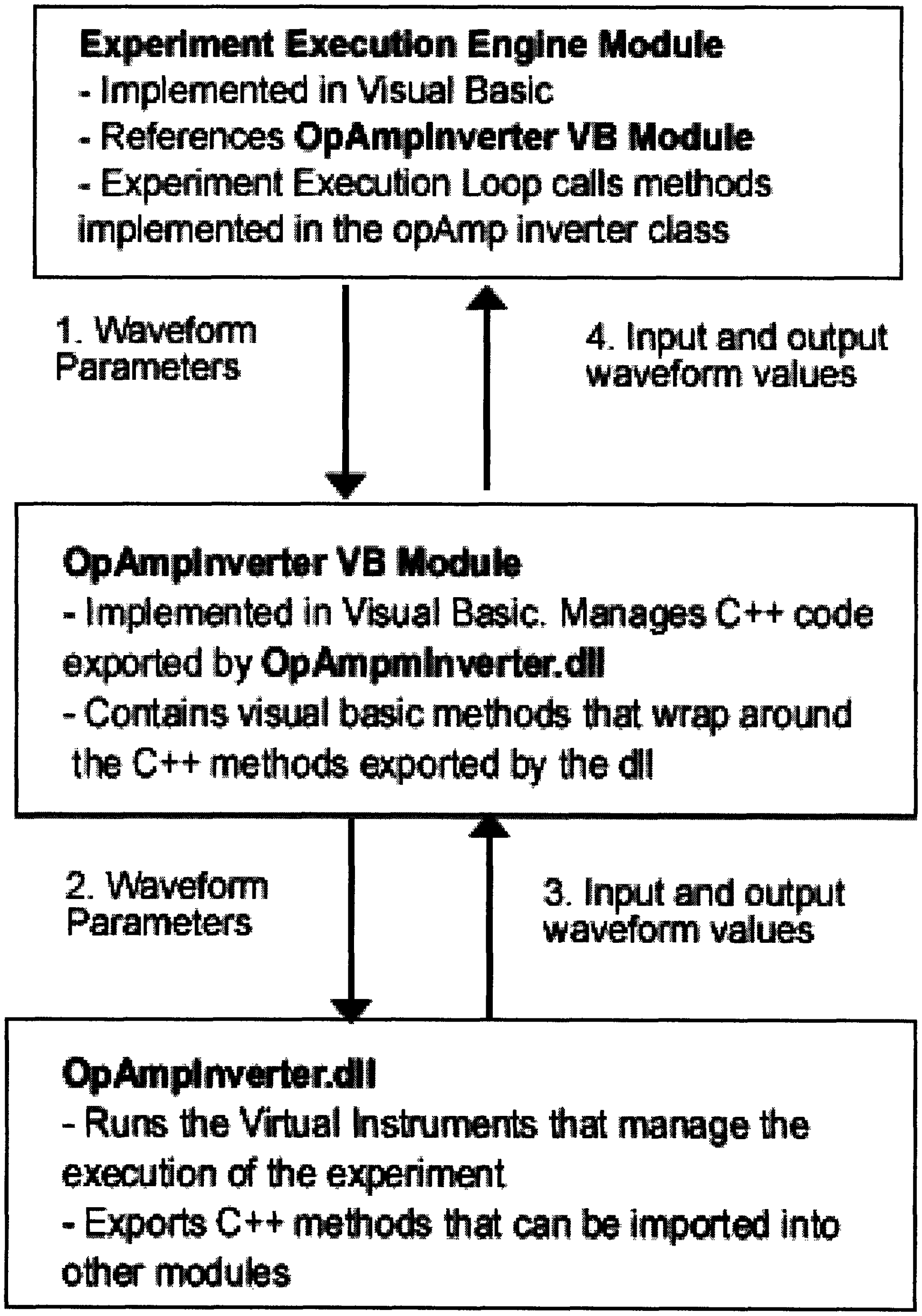

Experiment Execution Engine Module
- Implemented in Visual Basic
- References OpAmpInverter VB Module
- Experiment Execution Loop calls methods implemented in the opAmp inverter class
1. Waveform Parameters
4. Input and output waveform values
OpAmpInverter VB Module
- Implemented in Visual Basic. Manages C++ code exported by OpAmpmInverter.dll
- Contains visual basic methods that wrap around the C++ methods exported by the dll
2. Waveform Parameters
3. Input and output waveform values
OpAmpInverter.dll
- Runs the Virtual Instruments that manage the execution of the experiment
- Exports C++ methods that can be imported into other modules

Figure 3-3: This diagram shows how the experiment execution engine communicates with the ELVIS work station using virtual instruments. The numbered arrows indicate the flow of method calls and parameters from the experiment execution engine to the dll that conducts the experiment and back.

the ELVIS workstation and the results read back to the experiment execution engine.

5. An Experiments Result XML document that contains arrays of the time, input and output values is created and stored with the database record that corresponds to the current experiment. The status of the experiment is updated to indicate that it has been completed.

Lab Client

For the first version of the ELVIS weblab, a lot of the functionality involved in modeling experiments, displaying them to the user and collecting parameters from them are borrowed entirely from the Microelectronics weblab. In the Microelectronics weblab, the user specifies the parameters to be used when running an experiment by configuring signal measurement units (SMUs) that are connected to the parameter analyzer on the lab server side. These SMUs are displayed on the client as being connected to the image of the device being characterized. The client determines where to connect the SMUs to the device image by using information on the pixel location of their connection point sent by the lab server.

The ELVIS weblab uses the same approach to give users access to ELVIS instruments. In the first version of the ELVIS weblab, the user can specify the parameters to pass into the ELVIS Function Generator instrument (such as waveform type, frequency, amplitude and offset) by configuring an FGEN *port* that is connected to an image of the circuit being characterized. A comparison of the front ends of the Microelectronics and ELVIS v1 clients is shown in figure 3-4.

In order to specify experiments being sent to the lab server, the Microelectronics weblab used the notion of functions, which specified how each of the inputs to the SMUs would be applied. For example, some functions specified how voltages would be swept from a low to a high value at a given interval, while other functions encoded constant voltages. These functions were associated with respective SMUs and then encoded in the Experiment Specification XML document sent from the client to the

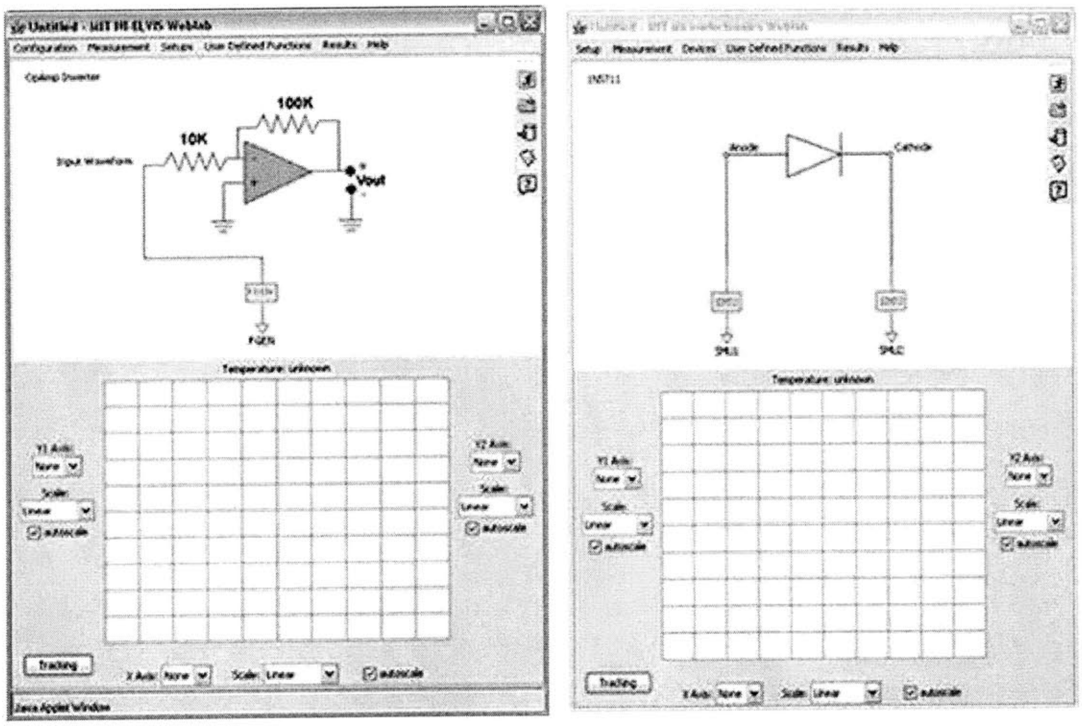
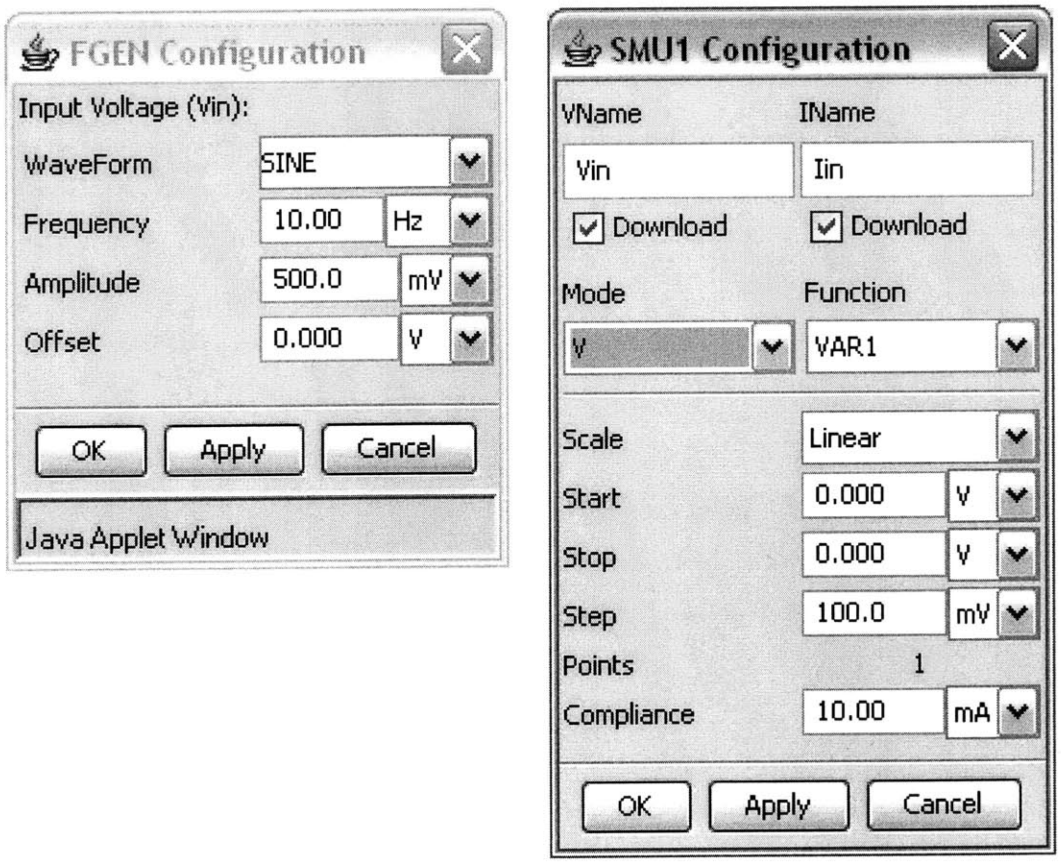


Figure 3-4: A comparison of the front end of the Microelectronics weblab client and the ELVIS weblab client. The striking similarity demonstrates how most of the concepts used to implement the Microelectronics weblab were recycled in the first version of the ELVIS weblab.

lab server. This document was enough to determine what values the lab server was to use when conducting the experiment.

For the case of ELVIS, we defined a new function type, whose parameters (type, frequency, amplitude and offset) specified the nature of the waveform to drive the opAmp circuit with. This function can be configured by the user on the client by clicking on the Function Generator instrument label displayed with the experiment. This action will bring up a dialog, which is similar to the one used to configure SMUs in the Microelectronics weblab. A comparison of the two dialog boxes is shown in figure 3-5.


FGEN Configuration

Input Voltage (Vin):

WaveForm	SINE	
Frequency	10.00	Hz
Amplitude	500.0	mV
Offset	0.000	V

SMU1 Configuration

VName	IName
Vin	Iin
✓ Download	✓ Download
Mode	Function
V	VAR1

Scale	Linear	
Start	0.000	V
Stop	0.000	V
Step	100.0	mV
Points	1	
Compliance	10.00	mA

Figure 3-5: A comparison of the dialogs used by the user to specify parameters for experiment execution for the Microelectronics weblab client and the ELVIS weblab client.

Strengths and weaknesses

By wholly adopting components of the Microelectronics weblab, the first ELVIS weblab implementation implicitly benefited from a wealth of software development work that had been ongoing for a number of years and which had introduced many great and necessary features for any online laboratory. This resulted in a feature rich lab server and client with minimum effort, eliminating the need to recreate features that have already been implemented and debugged.

The MIT approach also made it very easy to create new ELVIS based experiments without concerted software development efforts. All the functionality in the lab server and the client could be reused wholesome for new experiments, with the only modifications being required in the experiment execution engine.

Using LabVIEW to connect to ELVIS enables us to take advantage of the provided ELVIS source code, which gives us access to much of the functionality that comes with the ELVIS platform. The visual interface that comes with the LabVIEW programming environment also means that instructors do not need to be well versed in writing code in order to create and test new laboratory exercises. We were also able to utilize the processing, data manipulation and instrument communication capabilities of the LabVIEW API to enhance the execution of our experiments.

A major weakness of the MIT approach was the fact that only one experiment could feasibly be set up on the ELVIS work station at a time .

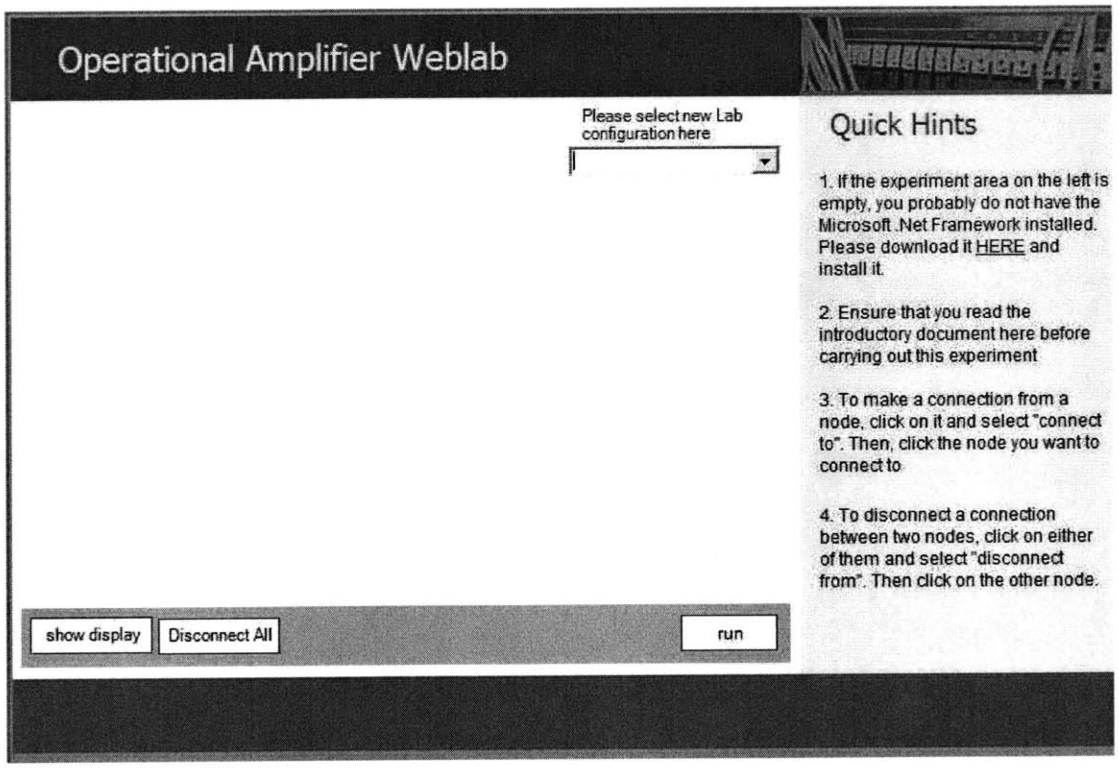
3.1.2 OAU Approach: The OpLab

The iLab team at OAU adopted a different approach in their development of the first version of the ELVIS iLab software, dubbed the OAU opLab. On the hardware side, they designed a circuit that uses a software-controlled switching matrix to allow the realization of up to six different operational amplifier configurations.

On the software side, their lab server and client implementations departs greatly from previous weblab implementations. They do away with the entire lab server as implemented in the Microelectronics weblab and implement an experiment execution engine as a stand alone module within the service broker. As a result, the lab lacks all the administrative functionality that was already implemented in the Microelectronics weblab lab server. This design decision was largely driven by the need to complete their implementation before a hard deadline.

Their experiment execution engine communicates directly with C++ dlls that come bundled with the ELVIS platform. As a result, they do not have to go through the route of creating custom LabVIEW virtual instruments and dynamic link libraries to communicate with the ELVIS board, as is the case with the MIT design.

The main innovation in opLab is their client implementation. Departing completely from the iLabs tradition of using java applets as clients, the OAU team implemented a C# form based client that can be accessed from a web browser on a windows machine. The user interface presents the user with a choice of the six implemented operational amplifier configurations and asked him/her to choose which one to analyze (see figure 3-6).


Operational Amplifier Weblab
Please select new Lab configuration here
show display
Disconnect All
run

Quick Hints

1. If the experiment area on the left is empty, you probably do not have the Microsoft .Net Framework installed. Please download it HERE and install it.

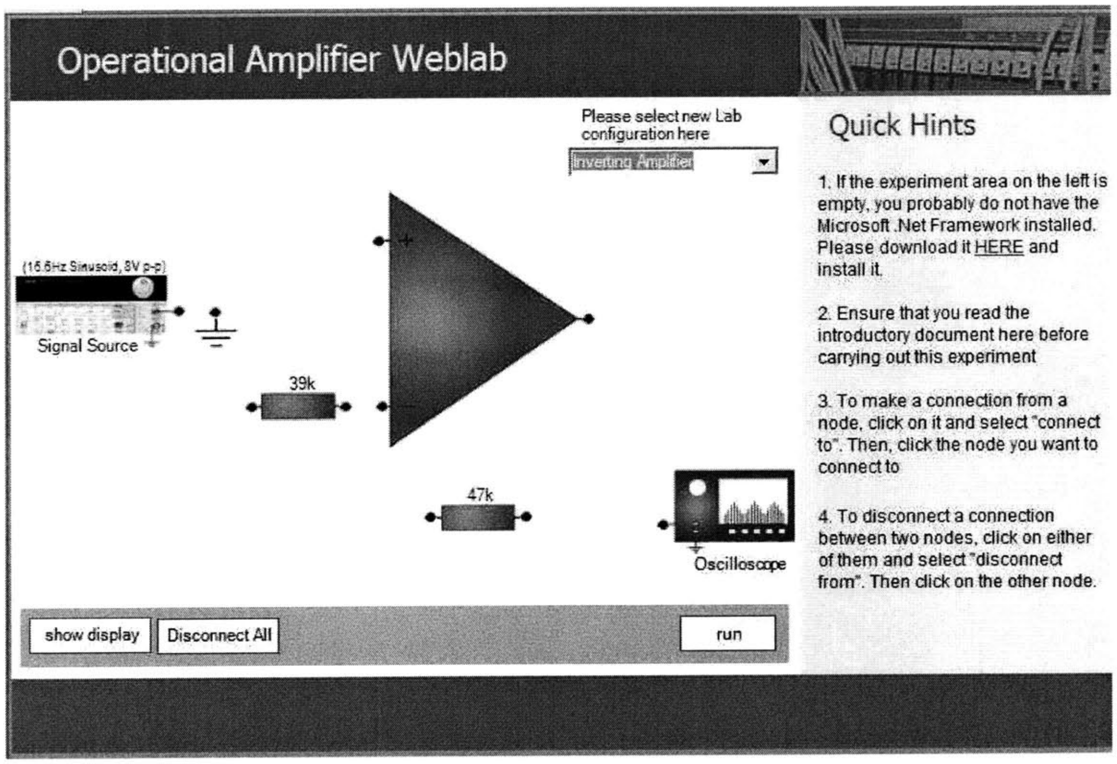
2. Ensure that you read the introductory document here before carrying out this experiment

3. To make a connection from a node, click on it and select "connect to". Then, click the node you want to connect to

4. To disconnect a connection between two nodes, click on either of them and select "disconnect from". Then click on the other node.

Figure 3-6: This page of the OAU opLab client presents the user with a choice of operational amplifier configurations.

Based on the users choice, the client selects from a set of available components and displays the ones necessary to realize that configuration as shown in figure 3-7. For example, the components displayed for an inverting configuration were different from those displayed for an integrator or differentiator. The client then requires the user to connect the different components in order to realize the configuration that he/she had chosen. Wrong connections are automatically detected and reported to the user.


Operational Amplifier Weblab
Please select new Lab configuration here
Inverting Amplifier
(16.6Hz Sinusoid, 8V p-p)
Signal Source
39k
47k
Oscilloscope
show display
Disconnect All
run

Quick Hints

1. If the experiment area on the left is empty, you probably do not have the Microsoft .Net Framework installed. Please download it HERE and install it.

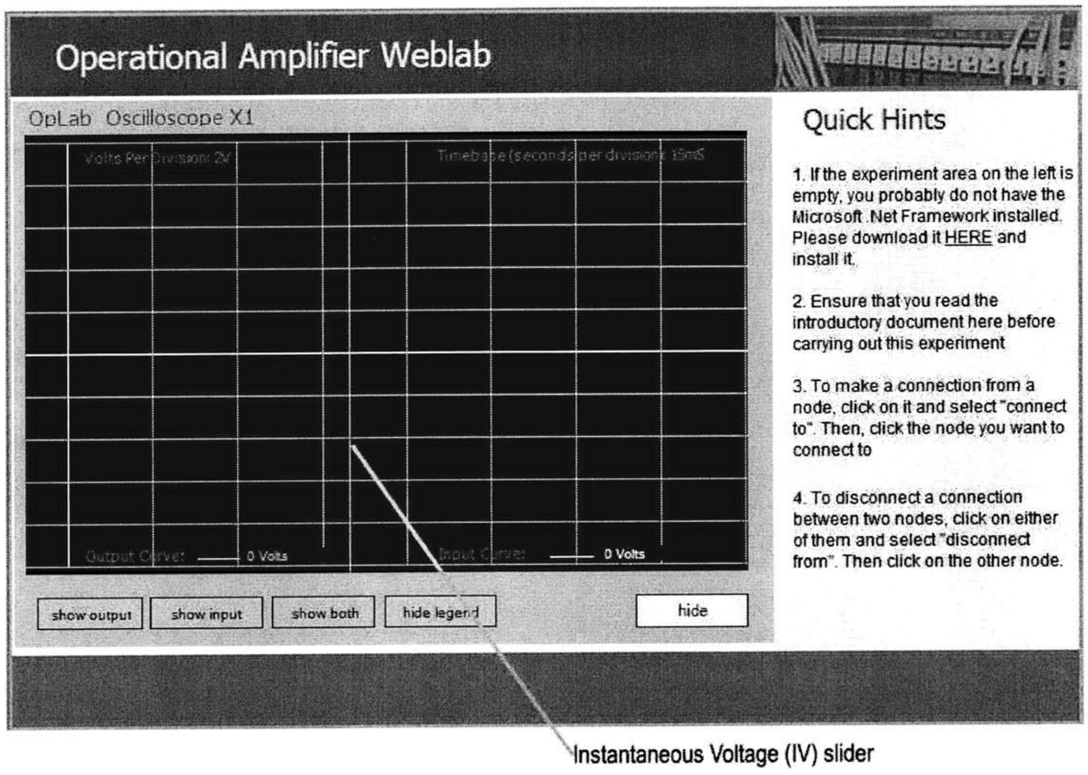
2. Ensure that you read the introductory document here before carrying out this experiment

3. To make a connection from a node, click on it and select "connect to". Then, click the node you want to connect to

4. To disconnect a connection between two nodes, click on either of them and select "disconnect from". Then click on the other node.

Figure 3-7: This page of the OAU opLab client presents the user with the components required to create an inverting opAmp configuration. The user is then required to figure out which components to connect together before submitting the experiment to the lab server for execution.

Once the right configuration is realized, the user can then submit the experiment to the lab server for execution. The results of running the experiment are displayed on an oscilloscope as shown in figure 3-8.



Quick Hints

1. If the experiment area on the left is empty, you probably do not have the Microsoft .Net Framework installed. Please download it HERE and install it.

2. Ensure that you read the introductory document here before carrying out this experiment

3. To make a connection from a node, click on it and select "connect to". Then, click the node you want to connect to

4. To disconnect a connection between two nodes, click on either of them and select "disconnect from". Then click on the other node.

Figure 3-8: This page of the OAU opLab client displaye the results of executing an experiment to the user

Strengths and weaknesses

The main strengths of the OAU approach lie in the great interactivity of its client. Requiring the user to connect the components adds great pedagogical value to the lab, ensuring that students in the most basic classes have a good understanding of which components are used to create the different operational amplifier configurations.

The OAU approach also solved the problem of having only one experiment at a time on the ELVIS work station by using a software-controlled switching matrix.

By stripping most of the lab server functionality, however, the opLab lost a lot of the functionality that had been added to previous weblabs over time, depriving it of important administrative functionality and experiment data management. The opLab also modifies the service broker, which is generally undesirable.

The design of the opLab also makes most of the client software very experiment specific, which means that a lot of code would need to be rewritten should a new experiment be set up. This can be avoided by specifying XML documents that determine how the client is to display experiments of this nature. In addition, implementing the client in C# means that only users running windows on their machines will be able to use it, which is undesirable.

Finally, the opLab did not give students the ability to manipulate the nature of the input signals used to drive the implemented circuit, a feature that would have added pedagogical value to their experience with the lab.

3.2 Motivation for the new software architecture

After comparing the approaches adopted by the two teams and presented in the previous section, we decided to come up with a unified approach that would help advance software development efforts for the iLab Africa project to the next level by leveraging the strengths demonstrated by both approaches.

From the relative ease with which the MIT approach adopted the microelectronics weblab software to work with the ELVIS work station, it is fair to suppose that the current weblab software architecture is a good starting point for ELVIS development.

This is because it implements many well thought out features that should be important to incorporate into any new weblabs that we develop and deploy in Africa, making them sustainable, manageable and usable by users in all parts of the world.

However, the implementation of these features means that the software itself is fairly complex and modifying it to support new labs can be quite tedious. This is a tradeoff that the OAU team faced during the development of the OpLab, which forced them to strip a lot of functionality from existing components. We also went through the same experience while developing software for the first version of the ELVIS weblab here at MIT.

Developing labs around ELVIS has also brought forth the observation that the existing user interface to the microelectronics lab is more suitable for purely analytical labs. This was especially demonstrated by the design of the OpLab and from my interaction with students and teachers at OAU. The consensus seems to be that pedagogical value could be added to the labs by having students actually wire up components of the circuit that they wanted to analyze to better simulate the actual experience with physical components and also stimulate the students by getting them to think more about the actual circuit connections. The OpLab builds on this observation and allows students to connect circuit components in order to create and analyze different circuit configurations. The software development efforts for this project will need to take this into consideration in order to exploit this new capability.

3.3 Specifications of a new design

Stemming from these observations, we came up with a new design that captures the essence of these "interactive" experiments, while speeding up the development of new labs. The proposed design will draw on the strengths of the opLab, which gives students a lot of flexibility when performing experiments, and from the strengths of the current microelectronics weblab architecture, which allows the easy, intuitive creation, management and deployment of new labs by instructors.

The design will provide a simple web based lab management interface that can be used by instructors to specify how the client will display different components and what connections are valid for a given configuration. This should eliminate the need to rewrite client software for every new lab and substantially simplify the process of new lab deployment.

The graphical user interface will allow students to add arbitrary components from a pre-determined set to a circuit configuration and create connections between these components. The components available to students will range from actual physical devices such amplifiers, diodes, transistors, resistors and capacitors to the wide range of virtual instruments that come bundled with ELVIS and other analytical devices connected to the lab server machine.

The design will also need to be transparent to the method that is used to communicate with the actual hardware. This means that ideally, only the experiment execution engine should contain hardware specific code. This will enable labs that communicate with instruments through proprietary software such as LabVIEW to co-exist harmoniously with those that talk to the hardware directly, such as the opLab.

The next chapter focuses on a more detailed specification and implementation of a unified approach that fulfils the goals outlined in this section.

Chapter 4

ELVIS Weblab v2: A unified Approach

This chapter describes the software architecture an ELVIS weblab (dubbed ELVIS weblab version 2) that unifies the ideas generated by the first two implementations of software to develop weblabs based on the ELVIS platform and the iLabs shared architecture described in chapter 3. The main feature of the new design is the support for a client that enables the manipulation of multiple electric components and ELVIS instruments by the user.

4.1 Architecture Overview

The architecture of ELVIS weblab v.2 is based on the iLabs batched software architecture. In particular, this ELVIS weblab extends a lot of the functionality that is implemented in the first version of the ELVIS weblab in order to satisfy the design specifications outlined in the last section of chapter 3. As with previous weblabs, the functionality of the ELVIS weblab is contained in three components: the client, the service broker and the lab server.

The ELVIS weblab clients main job is to display an experiment setup to a student and allow the student to interact with software representations of different electronic components by connecting them together and specifying input values to be used to drive the resulting circuit. The client for this lab is implemented as a java applet.

The service broker is a web service that mediates communication between the lab server and the client. The service broker specifies an interface which must be implemented by both the client and the lab server in order for them to communicate successfully. As with the previous implementation of the ELVIS weblab, there is no modification made to the service broker to support the new functionality in this weblab.

The job of the lab server is to receive experiment specifications from the client (through the service broker), validate the experiment and either perform the requested experiment or provide feedback if the specification is invalid. The lab server is implemented as a web service. It contains within it the experiment execution engine that communicates with the actual hardware and performs the experiment.

The main feature that distinguishes this version of the ELVIS weblab from the first version is its support for more user interaction on the client side. The client allows students to drag and connect electric components on a drawing canvas before sending the experiment specifications to the lab server for validation.

4.2 Design Process

4.2.1 Describing the Experiment Domain

In order to describe experiments conducted by students in software, we developed the model described below. This model is mainly motivated by the way in which users will be interacting with representations of various electric components on the client. Since the user will be able to manipulate electric components and ELVIS instruments, the model contains terms for describing the different types of electric components that can be used in an experiment.

Experiment Setup

The technical term used in the software to refer to a circuit that has been constructed on the ELVIS Board to conduct a single experiment is *Experiment Setup.* An *Experiment Setup* will contain attributes such as name, description and unique numerical ID. An *Experiment Setup* will also contain a list of *Electric Components* that are used to construct it.

Electric Component

Electric Component refers to any of the components used to create the circuit used in the experiment. Each *Electric Component* will have a name, a symbol, an icon associated with it and pixel information on the locations of the terminals that can be used to connect it to other components.

Electric components are further subdivided into two categories:

- ELVIS Instruments: *ELVIS Instrument* refers to any of the instruments included in the ELVIS instrumentation suite to source signals and analyze the output from circuits built on the prototyping board. The list of available ELVIS instruments includes a function generator, an arbitrary waveform generator, an oscilloscope, a multimeter, a digital reader/writer and a variable power source. The client will allow users to specify instrument parameters that will determine the nature of the signals used to drive the circuit, as well as the nature of the output. Besides inheriting the attributes of *Electric Components*, each instrument will also have associated with it a type and a list of its compliance values, such as maximum current, maximum voltage and maximum frequency.

- Component Profiles: *Component Profile* refers to any of the physical devices such as operational amplifiers, diodes, transistors, resistors, capacitors or inductors used to create the circuit in the experiment. Besides inheriting the attributes of *Electric Component*, each *Component Profile* will also have associated with it information on its specifications, such as resistance, inductance and capacitance, and a number that will distinguish it from other similar profiles.

Figure 4-1 below illustrates the relationship between the terms introduced above.

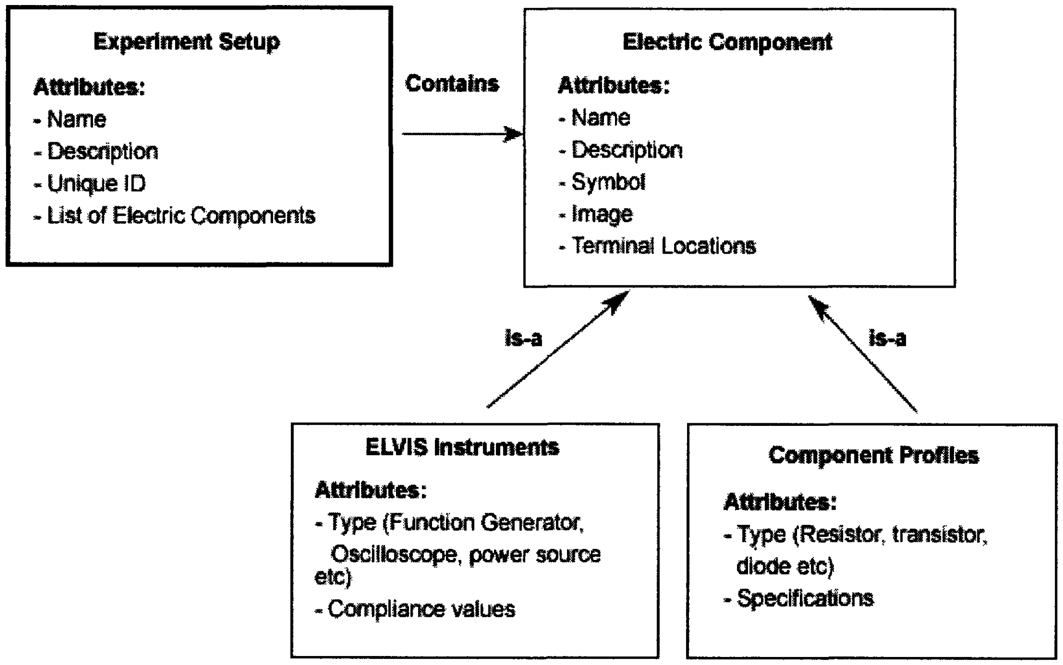


Figure 4-1: This figure shows the classification of circuit components that the user can interact with and their features. This model motivates the design of other lab server and client components.

4.2.2 XML Document Specification

The architecture of the lab server and client software is greatly influenced by the nature of the XML documents that will be used to specify the Lab Configuration (created by the lab server and sent to the client), the Experiment Specification (created by the client and sent to the lab server) and the Experiment Results (created by the client and sent to the lab server). In order to maximize code reuse, the XML documents are structured in such a way that client code can be reused across different experiments.

Lab Configuration XML Document

This document is created by the lab server. It contains information about which experiments that can be performed on the lab server at a particular time. This information is drawn from the lab servers database. Information contained in this document should be adequate for the client to display information on any experiment that is built on the ELVIS board. This ability will enable us recycle client code for different experiments.

The nature of this document captures the model that was described for experiments above. An example of the XML document that I used for this version of the ELVIS weblab is shown in the appendix A.

Experiment Specification XML Document

This document is generated by the client in order to specify the nature of the experiment to be conducted on the lab server. For this version of the ELVIS weblab, this document should contain the following information:

- The ID of the experiment which the user has submitted. This is the same ID that was assigned by the lab server in the Lab Configuration XML document.

- A list of all the component profiles that are used by the user in this experiment. For each component profile, the document should contain a list of all the terminals, each of which should contain a list of other terminals that it is connected to. This cascade of information will help recreate the circuit as it was connected by the user for validation.

- A list of the ELVIS instruments that are used by the user in this experiment. For each instrument, the document should not only contain a list of terminals and their connections, but also information on how the instrument has been configured by the user. Instruments are configured in the same way as in version 1 of the ELVIS weblab (see Page REFEFENCE).

An example of an Experiment Specification XML document is included in appendix A.

Experiment Result XML Document

This document is generated by the lab server upon the successful completion of an experiment. The document contains entries of data that correspond to each of the parameters that the user wants to measure for the given experiment. For a simple experiment that contains an input waveform, an output waveform and time values, the document would contain an array of double values for each of the three parameters. The clients graphing API would then use this information to recreate these waveforms for the user.

An example of an Experiment Result XML document is included in appendix A.

4.2.3 Lab Server

Besides storing information necessary to generate the Lab Configuration XML document, the lab server has an administrative structure that handles roles such as managing communication with the service broker, managing instruments and component profiles, viewing and analyzing pertinent server data such as web method activity and experiment queue status and access control management. Most of this functionality is already implemented in the first version of the ELVIS weblab, having been borrowed entirely from the Microelectronics weblab.

Only the device-specific sections of the first version of the ELVIS weblab lab server therefore needed to be redesigned for the second version. Since the lab server is implemented as a three tier web application, enhancing new functionality involves modifying all three levels:

- The database tables needed to be redesigned in order to store information that can be used to generate a lab configuration XML document of the nature described in the previous section.

- The backend code that interfaces the database to the actual lab server user interface needed to be changed.
- The user interface (.ASP files that the user interacts with) also had to change as a result.

Database Design

The database is the most important component of the lab server. It determines the nature of the information that can be stored in the application and hence to a large extent the extent of the functionality that can be built upon that information. Designing the device specific tables in the database requires a good understanding of how the experiments will be represented on the client side to the user and ensuring that all the necessary relationships are represented in the resulting table structure. It also requires examining the information to be exchanged carefully in order to determine what aspects can be hard-coded in the system, as opposed to being represented in the database.

Based on the experiment domain model presented earlier in this chapter, deciding which tables to create in the database was pretty straightforward. The first and most obvious table that was required for this lab was the *ExperimentSetup* table. This table stores general information about every experiment setup that is created on the ELVIS board. The necessary fields in this table include name and description, as well as a unique, auto-generated, numeric ID that identifies the experiment setup.

In order to store information about electric components, we decided to distinguish slightly between devices and instruments. We decided to hard-code most of the information about ELVIS instruments in the client, thereby reducing the complexity of the database structure in the lab server. Information about devices is stored in a series of tables that store the names, descriptions, icon file locations, terminal information and specification information for each device type, while instruments are all stored in a single table.

When a device type is created (for example a resistor), one entry is made in the

ComponentType table, detailing the name, description, symbol and icon location for that device type. Entries are also made in the terminals table (*ComponentTypeTerminalConfig*), indicating the pixel locations of all the terminals and their orientation in the icon. The last set of entries is made in the specifications table (*ComponentTypeSpecConfig*), indicating what specifications will be required for the component type (such as resistance in the case of a resistor).

It should be noted that a device type entry in the database does not correspond to an actual physical device. It is merely a template that will be used to determine what characteristics the actual devices will possess. The actual devices are referred to as *Component Profiles* in the system(see figure 4-1); they correspond to single entries in the *ComponentProfile table.* When the lab administrator creates a new experiment setup and adds a component profile to it, a single entry is made in this table.

A component profile refers to a specific instance of a component type, with the values for the required specifications filled in. An example of a component profile would be a 100KOhm resistor, or a 10microFarad Capacitor. One component profile needs to be created for each electric component to be used in the experiment being created. The *ComponentProfile* table stores the relationship between the generated unique ID belonging to that component profile and the experiment setup that it belongs to, as well as mappings to the specification values in the ProfileSpecs table that belong to it.

It is important to note, therefore, that it is not component types that end up being added to the experiment setups; it is component profiles.

Since we hard-coded a lot of information about ELVIS instruments in the client, this eliminated the need to store information such as name, description and image location for the different instrument types. Instead, we use a separate table named *Instruments*, similar to the *ComponentProfile* table, which stores the relationship between unique instances of an instrument used in an experiment setup and the experiment setup it belongs to. This table also contains fields that store compliance values detailing the limits to the values that can be applied to that instance of the instrument.

Just like in the case of component profiles, the lab server administrator will need to create a new record for each instance of an instrument to be used in the experiment.

When modifying the database to reflect the new table structure, we realized that database tables for the first version of the ELVIS weblab can be categorized into two broad categories: device specific tables that stored information about the actual devices on which experiments are conducted, and administrative tables, which store the information necessary to support functions such as managing communication with the service broker, viewing and analyzing data on web method activity and queue status, managing permissions and usage classes etc.

Only the first set of tables (device specific tables) needed to be modified in order to enable the lab server to work with the new lab configuration document. As a result of this observation, we decided to create a separate database that stores device specific information, a decision that will hopefully simplify the modifications necessary to enhance the functionality of this lab in future implementations.

The resulting database structure is shown in figure 4-2.

Once we created the database structure, the next step involved creating the set of procedures necessary to manipulate the information stores in these tables. These procedures are stored in the database (as SQL stored procedures) and they completely specify the process of creating and deleting experiment setups, component types, component profiles and instruments.

Back end and User interface

With the new database structure in place, modifying the rest of the Microelectronics weblab lab server back end to work with the new lab was pretty straightforward. The first step that we took was to create a separate visual basic module named *DeviceManager*, which provides the visual basic methods that interface with the stored procedures in our new database. This module exposes these methods to the User Interface layer (implemented using .ASP), enabling lab server users to perform all the operations necessary to manipulate the stored data from the lab servers web interface.

Besides creating the *DeviceManager* module, we also modified the code in the

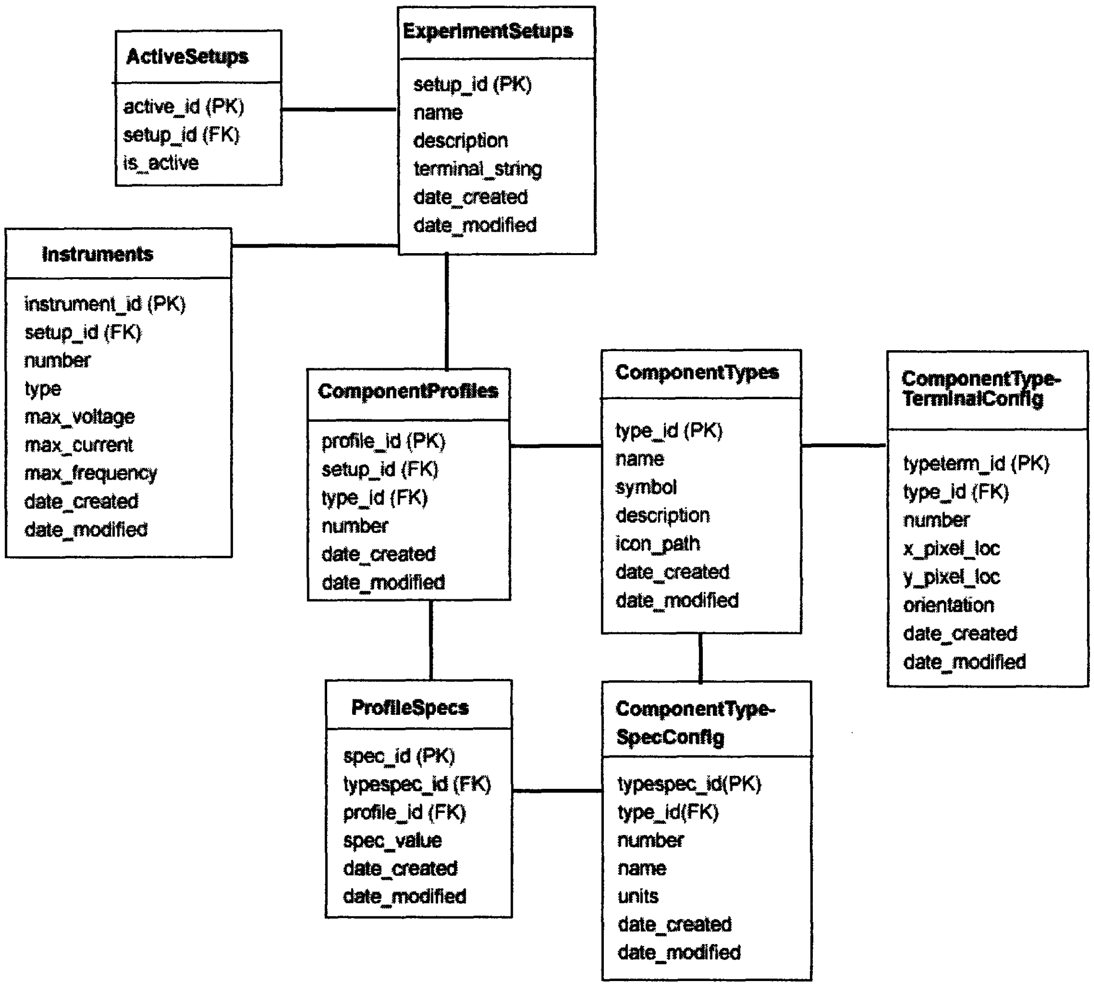


Figure 4-2: This figure shows the database structure used to store information about experiment setups for ELVIS weblab v2. Note: In the diagram, PK refers to Primary Key while FK refers to Foreign Key.

Experiment Validation Module, whose job it is to examine the specifications passed in by a student performing an experiment and ensure that they are compliant with the corresponding experiment setup on ELVIS. Besides checking to ensure that compliance values for all the configured instruments are met, this module also ensures that the terminals in all the components are connected in the right manner.

Most of the other modules in the back end of the ELVIS weblab V.1 are completely device independent and are therefore incorporated unmodified into ELVIS weblab v.2.

User Interface

The user interface for the lab server for v.1 gives the user to all the functionality described in the previous section via a secure, login requiring, web interface. This interface is implemented as a .NET web application using .ASP. The .ASP pages import the back end modules described previously and make calls to them in order to interact with the underlying SQL database.

Since most of the back end functionality of the v.1 lab server can be imported wholesome into the v.2, the same case applies to most of the .ASP pages in the front end. The only sections that needed to be redesigned and re-implemented were the pages that managed the creation and editing of experiment setups.

The v.2 lab server has two .ASP pages that manage the creation and editing of experiments on ELVIS. The first page allows users to create, edit and delete component types, while the second page allows users to create, view and edit experiment setups.

Each of the pages operates in one of two modes. In the first mode, the page displays a listing of all the experiment setups or component types that exist in the system, with a brief summary of each entry. This mode also allows the user to delete existing entries, or create new entries. The second mode allows the user to edit a specific experiment setup or component type. For experiment setups, this page also allows the user to instruments and component profiles of different types.

One feature of the User Interface that is not yet implemented is an allowance for users to specify the terminal connections that will lead to a valid experiment setup through the web. Currently, the user has to enter all the terminal connections for a valid configuration by hand, which can be tedious for an elaborate experiment.

Experiment Hardware and Execution Engine

One aspect of v.2 that was not implemented at all is the switching matrix. This was mainly due to delays in securing the necessary hardware from National Instruments. The goal of infusing the switching matrix is to replicate the desirable characteristics of the OAU opLab experiment discussed in Chapter 3, which allows multiple circuits to be built on the ELVIS board at the same time. Future versions of the ELVIS weblab will incorporate this aspect, hence increasing the options offered by the Experiment Execution Engine.

As a result, the hardware setup and the experiment execution engine used in v.2 are identical to those used in v.1, which are described conclusively on chapter 3 of this document.

4.2.4 Lab Client

The client implemented for v.2 is based largely on v.1 client, which is in turn based on the Microelectronics weblab client. The client is therefore implemented as a Java applet that can be loaded from any Java-enabled web browser. The client handles experiments based solely on the information passed in through the lab configuration XML document. It enables users to select from a list of provided electric components and connect then on a drawing canvas.

There already existed a robust back end for the client in the Microelectronics weblab and in v.1 of the ELVIS weblab. This back end supports many features that are desirable for any online laboratorys client, such as saving experiment specifications, graphing results, loading saved experiment specifications, defining user-defined functions and tracking experiment progress. Furthermore, the existing client software cleanly manages communication between the client and the lab server regardless of the user interface. As a result, a lot of code in the existing client can be reused in the new client design.

The functionality of the weblab client is implemented using 5 java packages. Three packages are imported wholesale from v.1. These are the graphing package, which handles the back end that supports the display of experiment results, the xml package that handles parsing of XML documents and the server interface package that handles messages to and from the service broker using the SOAP standard.

The rest of the weblab functionality is implemented in a back end package named client that manages and manipulates the object model for experiments displayed to the user, and a front end package named graphicalUI that handles all the aspects of displaying experiments to the user and displaying the results of the users input.

In order to implement a client that handles experiments specified using the new experiment model, we designed classes that belong to the client package, and which correspond to the main components of our experiment model. In particular, we designed classes that model experiment setups, electric components, component profiles, component types and instruments. The relationships between these different classes mirror their relationships in the experiment model presented earlier in this chapter.

In addition to these classes, we also reused classes representing concepts that were already present in v.1, such as terminals, functions, experiment specifications, lab configurations and a weblab client class that manages the interaction between the graphical layer and the underlying object model.

The front end of the client was also radically redesigned to enable clicking, dragging and connecting electric components, something that was completely absent from earlier weblab implementations. When the user chooses the experiment to conduct, the new client presents the user with a blank drawing canvas. On right clicking anywhere on this canvas, the user is presented with a menu that contains all the component profiles and instruments that are associated with the current experiment setup. The user can then select from amongst these components and drag them from the menu onto the drawing canvas. The user can place the selected component anywhere on the drawing canvas.

In order to simplify the first implementation, we introduced the concept of a special component known as a node. A node allows the user to connect components. The user cannot make these connections directly; components can only be connected to a node. This greatly simplified the drawing algorithms required to keep track of different components and their connections.

Once the user is satisfied with the established connections, he/she can then submit the experiment to the lab server for execution. When the user does this, the client will generate an Experiment Specification XML document (described earlier in this chapter) and send this to the lab server through the service broker. The results of experiments submitted by this client are handled in exactly the same way as in the version 1 of the ELVIS weblab.

The entire front end of the client was implemented using javas graphic packages (swing and AWT).

4.3 Results

The first prototype of the system described in this chapter has already been fully implemented. This first run is mainly a proof of concept; most of its functionality of not yet ready to be deployed.

For the purpose of demonstrating the system, the implemented software has been successfully used to manage a simple operational amplifier experiment that accepts a user-specified waveform as its input, inverts it, and displays both the input and the resulting output waveform on the client.

Besides fixing any latent bugs, a number of enhancements can be made to the system based on initial interactions with it.

For the lab server:

- This was discussed earlier, but it is worth mentioning again that adding a switching matrix to the ELVIS hardware could enable support for multiple experiments on the same board, which would add a lot of value to the implemented weblab.

- The validation module could be given more deductive ability, thereby enabling it to give more constructive feedback to the user when an experiment specification is invalid. For example, the engine should be clever enough to realize when the user creates a non-inverting configuration as opposed to an inverting configuration as is required by the experiment, instead of simply lumping all errors into one category.

- The LabVIEW dlls that execute the actual experiment could be improved to enhance their error detection capabilities, thereby increasing their tolerance for extreme inputs from users.

- The .ASP page that manages editing experiment setups could be improved on to give instructors an easier way of specifying the correct configuration for a given experiment. A suggestion that is explored in this chapter is having an applet popup that duplicates the clients functionality, enables the instructor connect the components as should be done by students and stores a string with the resulting configuration.

For the client:

- A lot of work could be done in cleaning up the user interface. The current implementation is largely a proof of concept that used the first idea that came to our mind and may not be adequate for a distribution version.

- We also noted that the size of the client increased considerably when the new functionality was incorporated, something that should be guarded against by exploring other software design patterns and data models that can support the same functionality with less overhead.

- Currently, the functionality for saving experiment specifications and loading saved specifications from the service broker is disabled. This is because supporting it would require storing information on how components are laid put and connected on the board, which we have not explored yet. Such an enhancement that would greatly improve the usability of future releases of the client.

Chapter 5

Conclusion and Future Work

Conclusion and future work Our experiences with the first two versions of the ELVIS software demonstrate the enormous potential of the ELVIS platform to support weblabs that span the entire range of the Electrical Engineering and physics curriculum. The first implementations have also shown a reliable model that can be used to bring various ELVIS instruments online and enable students to interact with them and conduct experiments. The existing ELVIS weblab implementations therefore represents a solid foundation on which to base future weblabs, which should exercise more ELVIS instruments and give users more control over the nature of the experiments being conducted.

The experience of writing software to bring the ELVIS equipment online should also herald efforts to bring other devices in developing countries online and share labs amongst different institutions. This will make previously inaccessible devices available to students, thereby enriching their learning experience.

5.1 Future iLab-Africa software development efforts

The process of developing software for the ELVIS weblabs has been a great learning experience in software design experience and project management for those involved.

The breadth and depth of the tools and software skills involved has tested and developed our capabilities immensely. It has also been a unique experience because of the collaboration between the group at MIT and developers in Africa.

A few lessons have been learnt in the course of our one-year software development collaboration with the team at OAU, some of which apply generally to the development of iLabs in developing countries.

During our collaboration, one enhancement that was missing was a clear channel for communicating software specifications, current progress and actual code between the two groups. As the software around ELVIS becomes more complex and well defined, we at MIT will need to engage our colleagues more strongly in the development process, to the extent that we can hand over the management of lab software to them. Stronger interaction will also enable the development of complementary software components that can interface cleanly with each other, thereby distributing the software development load.

In view of the resources required to develop and deploy weblabs under the current software architecture, the future of iLabs in developing countries will most likely hinge on solutions that use less proprietary software products and fewer man hours. The shortage of competent software developers willing to handle projects such as this one and the high cost of necessary software tools might hinder the addition of necessary features to future weblab implementations, or discourage other departments/institutions from embracing the iLabs concept.

In order to counter this, the software architecture of future weblabs will probably need to contain a generic pluggable module that can be incorporated into any lab server/client, leaving lab administrators to bear a very small subset of the software development load. Future implementations of the lab server may also need to look into solutions implemented in non-proprietary software (currently .NET and Windows SQL server) in order to lower the costs and encourage more departments to put their labs online.

In parallel with these efforts, the cost of the hardware used to support iLabs in developing countries may also need to be trimmed. In fact, there are already ongoing efforts to develop a cheap parameter analyzer that can handle a range of experiments commonly used in introductory level Electrical Engineering classes. These efforts will need the support of similarly inclined software development efforts in order to make them effective in developing countries.

Appendix A

XML Specification Documents

A.1 ELVIS Weblab v1

A.1.1 LabConfiguration.xml

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE labConfiguration SYSTEM
  "http://ilab-labview.mit.edu/LabServer/xml/labConfiguration.dtd">
<labConfiguration lab="MIT ELVIS Weblab" specversion="0.1">
<setup id="5">
<name>OpAmp Differentiator Circuit</name>
<description>Aren't opAmpss just swell?</description>
<imageURL>http://localhost/labServer/setupImages/opAmpDifferentiator.gif</imageURL>
<terminal instrumentType="FGEN" instrumentNumber="1">
<label>Input Waveform</label>
<pixelLocation>
<x>121</x>
<y>94</y>
</pixelLocation>
<maxVoltage>2.0</maxVoltage>
<maxCurrent>0.1</maxCurrent>
```

```
<maxFrequency>1000</maxFrequency>
</terminal>
<terminal instrumentType="SCOPE" instrumentNumber="2">
<label>Oscilloscope</label>
<pixelLocation>
<x>195</x>
<y>156</y>
</pixelLocation>
<maxVoltage>2.0</maxVoltage>
<maxCurrent>0.1</maxCurrent>
<maxFrequency>1000</maxFrequency>
</terminal>
<maxDataPoints>500</maxDataPoints>
</setup>
</labConfiguration>
```

A.1.2 ExperimentSpecification.xml

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE experimentSpecification SYSTEM
  "http://ilab-labview.mit.edu/labServer/xml/experimentSpecification.dtd">
<experimentSpecification lab="MIT NI-ELVIS Weblab" specversion="0.1">
<setupID>1</setupID>
<terminal instrumentType="FGEN" instrumentNumber="1">
<vname download="true">VIN</vname>
<iname download="true">ID</iname>
<mode>V</mode>
<function type="WAVEFORM">
<waveformType>SINE</waveformType>
<frequency>100</frequency>
<amplitude>0.5</amplitude>
```

```
<offset>0.1</offset>
</function>
<compliance>0.1</compliance>
</terminal>
<userDefinedFunction>
<name download="true">SQRTID</name>
<units>A</units>
<body>SQRT(VIN)</body>
</userDefinedFunction>
</experimentSpecification>
```

A.1.3 ExperimentResult.xml

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE experimentResult SYSTEM
  "http://ilab-labview.mit.edu/labServer/xml/experimentResult.dtd">
<experimentResult lab="MIT NI-ELVIS Weblab" specversion="0.1">
<datavector name="TIME" units="s">0 0.002 0.004 0.006 0.008</datavector>
<datavector name="VIN" units="V">-0.615234375 -0.6005859375 -0.581054687
                                 -0.556640625 -0.5224609375 </datavector>
<datavector name="VOUT" units="I">6.1474609375 5.9912109375 5.8203125
                                  5.56640625 5.2294921875 </datavector>
</experimentResult>
\end{figure}
\section {ELVIS Weblab v2}
\subsection{LabConfiguration.xml}
\begin{verbatim}
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE labConfiguration SYSTEM
  "http://ilab-labview.mit.edu/LabServer/xml/labConfiguration.dtd">
<labConfiguration lab="MIT NI-ELVIS Weblab" specversion="0.1">
```

```
<experimentSetup setupId = 1>
<name>Inverting Amplifier</name>
<description>Characterizing an operational amplifier in the inverting configuration.
</description>

<!-- component types -->
<componentType typeId = 1>
<name>Resistor</name>
<symbol>R</symbol>
<description>Resistor</description>
<imageURL>http://ilab-labview.mit.edu/LabServer/images/components/Resistor.gif
</imageURL>
<terminal number = 1>
<pixelLocation>
<x>10</x>
<y>25</y>
</pixelLocation>
<orientation>LEFT</orientation>
</terminal>
<terminal number = 2>
<pixelLocation>
<x>40</x>
<y>25</y>
</pixelLocation>
<orientation>RIGHT</orientation>
</terminal>
<profile profileId = 1 profileNumber = 1>
<spec name = "Resistance">
<specValue>100 KOhms</specValue>
</spec>
```

```
</profile>
<profile profileId = 2  profileNumber = 2>
<spec name = "Resistance">
<specValue>10 KOhms</specValue>
</spec>
</profile>
</componentType>

<componentType typeId = 13>
<name>OpAmp</name>
<description>Operational Amplifier</description>
<imageURL>http://ilab-labview.mit.edu/LabServer/
         componentIcons/amplifier.gif</imageURL>
<terminal number = 1>
<pixelLocation>
<x>12</x>
<y>10</y>
</pixelLocation>
<orientation>LEFT</orientation>
</terminal>
<terminal number = 2>
<pixelLocation>
<x>12</x>
<y>40</y>
</pixelLocation>
<orientation>LEFT</orientation>
</terminal>
<terminal number = 3>
<pixelLocation>
<x>12</x>
```

```
<y>25</y>
</pixelLocation>
<orientation>RIGHT</orientation>
</terminal>
<profile profileNumber = 1 profileId = 3></profile>
</component>

<!-- Instruments -->
<instrument type = "FGEN" instrumentID = 1>
<maxVoltage>2.0</maxVoltage>
<maxCurrent>0.1</maxCurrent>
<maxFrequency>1000</maxFrequency>
</instrument>
</experimentSetup>
</labConfiguration>
```

A.1.4 ExperimentSpecification.xml

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE experimentSpecification SYSTEM
  "http://ilab-labview.mit.edu/labServer/xml/experimentSpecification.dtd">
<experimentSpecification lab="MIT NI-ELVIS Weblab" specversion="0.1">
<setupID>1</setupID>
<componentProfile id = 1>
<terminal number = 1>
<connString>4:5:6:8</connString>
</terminal>
<terminal number = 2>
<connString>5:8</connString>
</terminal>
</componentProfile>
```

```
<componentProfile id = 3>
<terminal number = 4>
<connString>1</connString>
</terminal>
<terminal number = 5>
<connString>1:2</connString>
</terminal>
</componentProfile>

<instrument type = FGEN number = 1>
<terminal number = 1>
<connstring>5</connstring>
</terminal>
<vname download="true">VIN</vname>
<function type="WAVEFORM">
<waveformType>SINE</waveformType>
<frequency>100</frequency>
<amplitude>0.5</amplitude>
<offset>0.1</offset>
</function>
<compliance>0.1</compliance>
</instrument>

<instrument type = SCOPE number = 2>
<terminal number = 1>
 <connstring>8</connstring>
</terminal>
<vname download="true">VOUT</vname>
</instrument>
```

```
<userDefinedFunction>
<name download="true">SQRTID</name>
<units>A</units>
<body>SQRT(VIN)</body>
</userDefinedFunction>

</experimentSpecification>
```

A.1.5 ExperimentResult.xml

ELVIS weblab v2 uses the same document as v1.

Appendix B

Report of visit to OAU

Samuel Gikandi 2/13/2006

Report on visit to Nigeria for the iLabAfrica project

This past January, I spent three weeks at the Obafemi Awolowo University in Ife Nigeria, working with the iLabs group there as part of the MIT iLab Africa project.

I was able to achieve most of the goals that I outlined in my proposal for the trip. My first goal was to discuss with the software development team at OAU on how to approach the integration of the NI ELVIS board into the iLabs shared architecture. I was able to work closely with Jonah (lead developer) at OAU towards the end of my trip and I demonstrated the features of my approach as well as learning about their implementation of an OpAmp characterization iLab. This discussion was especially fruitful because our approaches were different, with each approach having its strengths and weaknesses. We discovered that we could each borrow aspects from our approaches and will be collaborating on a unified design for future lab deployment.

I will briefly outline what I learnt about the development of the OpAmp lab at OAU for their first deployment. The starting points for developing this lab were the weblab components, namely Jims lab server and the java applet client. From these, the team at OAU stripped most of the functionality from the lab server and developed an entirely new client for their OpAmp lab. This was mainly necessitated by the need to develop this first version rapidly, but it also meant that the finished lab has lacked a lot of the structure that has been built into weblab over time. To assist with this, I have been able to modify the weblab server for the purposes of ELVIS. The only problem is that my lab server works with a different approach of communicating with the ELVIS board. Instead of talking to the board directly as the team at OAU does (using C++ calls to dlls that come with ELVIS), I create my own dlls using labview and talk to ELVIS virtual instruments. From our discussions, we decided that the team at OAU will look into the labview approach, because I was able to show that it speeds up the process of deploying new labs.

One of the main strengths of the OAU approach was the use of switches, which allowed them to have a large number of circuits on the same ELVIS board. Their first implementation allows users to characterize six different OpAmp configurations. Due to this observation, I will be working on integrating a switching mechanism into future lab server implementations.

Another great strength is the client design, which allows students to connect components to each other, as opposed to simply running experiments on an already constructed circuit. This gives students more intuition on connecting components, while giving feedback when students wire the components wrongly. The feeling was that this greatly improves the pedagogical value of online labs, especially those developed for introductory level classes.

The OAU client is however implemented using C# and this limits its use to windows platforms only. This is something that we agreed to move away from, given that the main reason they used C# was because they were more comfortable with C# than java. Towards this end, I have been working on developing a java client that borrows from the pedagogical strengths of the C# client, and hence makes it available on all platforms.

I should note here that my discussion with the OAU team was hampered by the fact that members of the software development team, whom I was to work with closely, were extremely busy with administrative and teaching work at the university. I nonetheless took advantage of the free time I found myself with to put final touches on an implementation of an ELVIS lab and client that maximizes the reuse of weblab components. I was also slightly limited in my implementation by the inability to connect my ELVIS board to my laptop, because I could not get the PCI card necessary for this in good time. I nonetheless made substantial progress with my work, because my approach is more Labview specific, as opposed to ELVIS specific.

During the three weeks I spent in Nigeria, I was also able to appreciate the limitations within facing the team at OAU when developing iLabs. One of the main limitations was manpower, as the people who were developing software for the labs had heavy administrative and lecturing responsibilities. Piotr and I suggested to them the possibility of hiring students to assist in developing new labs, which they did towards the end of the trip. However, the students who were assigned to developing lab software may not have enough time to understand and develop software based on the iLabs shared architecture, mainly because they are final year students hoping to graduate by June this year. This is clearly not enough time for them to make enough headway in understanding and modifying existing iLabs components. We did suggest to them the possibility of hiring students at freshman/sophomore levels and Prof. Kehinde promised to look into this.

It was nonetheless very clear that online labs would add a great deal to the academic experience of students. We were able to appreciate this on touring the campus, where we saw the immense shortage of equipment for conducting experiments that iLabs would alleviate. This shortage is especially acute for students in the introductory level classes, which tend to be extremely large. Luckily, the ELVIS board is especially useful for conducting experiments based on the curriculum in these early classes.

We also noted that the internal network connection at the university was extremely fast, and bandwidth should therefore not be a big limitation when considering locally deployed solutions for the university. Nonetheless, most students do not have access to private computers, a factor that may necessitate the design of an extremely intuitive client for these students to minimize the time spent at public workstations (which they sometimes have to pay for).

All in all, I was greatly impressed by the enthusiasm with which the team at OAU has adopted the iLabs concept and the efforts taken to infuse iLabs into the university curriculum. This was especially manifested in seminars organized to introduce faculty and students at the university to the concept of iLabs. Many ideas were brought up and discussed at these seminars and they really gave us a good idea of the attitude towards online laboratories at the university and the design decisions that would make them more effective for classes at the university.

Based on the visit in January, I will be engaging my colleagues at OAU further to exploit the wide array of virtual instruments that come with ELVIS. This will begin with the development of a new lab for logic gate characterization, which is what the group at OAU is currently working on. We will also explore more ELVIS functionality using new labs, to provide a basis for building future labs more easily.

Another project that will be immensely significant will be integrating Piotrs mini into the iLabs shared architecture, since it was so warmly received at the university by both faculty and students. This will open up the integration of a wide array of cheap labs into the iLabs shared architecture, engage students more intensely with the development of labs and thereby enrich the effectiveness of iLabs in Africa.

I also hope to engage colleagues in Africa more closely with the development of lab server and client software because it was apparent that they had difficulty customizing what already exists for their own needs. This is based on the observation that most of the features of the current implementation are well thought out and crucial for the deployment of labs that can be reliably administered to students.

Finally, there was the shared concern that there was not much communication between African universities involved in iLabs. The team at OAU did not communicate regularly with the teams in Uganda and Tanzania, although they felt this exchange of ideas would be beneficial for the future of iLabs in Africa. After the trip, I also noted that the concept of iLabs was now spreading to other countries (namely Kenya and Zambia) and it would be a great step forward if we can take advantage of the advanced involvement and understanding of the concept at OAU to establish a hub for the further deployment of iLabs at Africa. Indeed this would be a great springboard for further collaboration amongst African universities, which could only enrich the experiences of students at these institutions further.

Bibliography

[1] Phil Bailey. The ilabs shared architecture and the future of web-based laboratory experiments. [online]. available: http://como.cheng.cam.ac.uk/pdfs/workshop/phil_bailey_ws.pdf.

[2] J. del Alamo, J. Harward, S. Lerman, and K. Amaratunga. A web-service architecture to bring labs online. available: https://faculty.university.microsoft.com/2004/. Microsoft Faculty Summit, August 2004.

[3] J.A. del Alamo. Realizing the potential of ilabs in africa. *MTL Annual Research Report*, 2005.

[4] NI Educational Laboratory Virtual Instrumentation Suite (NI ELVIS). National instruments, (date downloaded: August 1, 2006) . [online].available:http://zone.ni.com/zone/jsp/zone.jsp.

[5] J. Harward et al. ilab: A scalable architecture for sharing online experiments. Gainesville, Fl, October 2004. ICEE.

[6] J. Harward et al. ilab: A scalable architecture for sharing online experiments. Gainesville, Fl, October 2004. ICEE.

[7] R.M. Fleder and R.Brent. Designing and teaching courses to satisfy the abet engineering criteria. *Journal of Engineering Education*, 92(1):7–25, 2003.

[8] Jim Hardison. Increasing reliability, reusability and measurement flexibility in the microelectronics weblab.

[9] J. Harward. Service broker to lab server api. (date downloaded: August 1, 2006). [online]. available: http://icampus.mit.edu/ilabs/architecture/downloads/default.aspx.

[10] iCampus. icampus: The mit-microsoft alliance. (date downloaded: August 1, 2006). [online].available: http://icampus.mit.edu.

[11] MIT iCampus. ilab: Remote online laboratories, (date downloaded: August 1, 2006) . [online].available:http://icampus.mit.edu/projects/ilabs.shtml.

[12] National Instruments. Creating and calling labview dlls with various data types from microsoft visual basic, (date downloaded: August 1, 2006) . [online].available: http://zone.ni.com/zone/jsp/zone.js.

[13] National Instruments. Elvis: Instrumentation, data acquisition, and prototyping for labs, (date downloaded: August 1, 2006). [online]. available:http://sine.ni.com/nips/cds/view/p/lang/en/nid/13137.

[14] National Instruments. Labview: The software that powers virtual instrumentation, (date downloaded: August 1, 2006).[online]. available:http://www.ni.com/labview/.

[15] National Instruments. Universities using nielvis. (date downloaded: August 1, 2006). [online].available: http://www.ni.com/academic/ni_elvis/universities_using_nielvis.htm.

[16] National Instruments. Why should i use labview in education?, (date downloaded: August 1, 2006) . [online].available:http://zone.ni.com/zone/jsp/zone.js.

[17] Microelectronics Techology Lab. ilab in africa: Overview. (date downloaded: August 1, 2006). [online].available: http://www-mtl.mit.edu/ alamo/ilaboverview.htm.

[18] Microsoft. Microsoft .net homepage (date downloaded: August 1, 2006) . [online]. available: http://www.microsoft.com/net/default.mspx.

[19] Chandan Nair. Enriched science and engineering education using the educational laboratory virtual instrumentation suite (elvis): Sharing successes across asean. available: http://www.cdtl.nus.edu.sg/link/nov2004/lit2.htm.

[20] Massachusetts Institute of Technology. Microelectronics weblab. (date downloaded: August 1, 2006) . [online].available:http://weblab.mit.edu.

[21] Geraldo Viedma. Design and implementation of the feedback systems web laboratory. Master's thesis, Massachusetts Institute of Technology, 2005.

[22] Gerardo Viedma, Isaac J. Dancy, and Kent H. Lundberg. A web-based linear-systems ilab. Portland Ore, June 2005. American Control Conference.

[23] The World Wide Web Consortium (W3C). Soap specifications. (date downloaded: August 1, 2006). [online]. available:: http://www.w3.org/tr/soap/.

[24] D. Zych. Lab client to service broker api. (date downloaded: August 1, 2006). [online]. available: http://icampus.mit.edu/ilabs/architecture/downloads/default.aspx.