iLab: A Scalable Architecture for Sharing Online Experiments

Authors:

V. Judson Harward, Massachusetts Institute of Technology, Cambridge, MA, jud@mit.edu
Jesús A. del Alamo, Massachusetts Institute of Technology, Cambridge, MA, alamo@mit.edu
Vijay S. Choudhary, Massachusetts Institute of Technology, Cambridge, MA, vijayc@mit.edu
Kimberley deLong, Massachusetts Institute of Technology, Cambridge, MA, kirky@ceci.mit.edu
James L. Hardison, Massachusetts Institute of Technology, Cambridge, MA, hardison@alum.mit.edu
Steven R. Lerman, Massachusetts Institute of Technology, Cambridge, MA, lerman@mit.edu
Jedidiah Northridge, Massachusetts Institute of Technology, Cambridge, MA, jedidiah@mit.edu
Daniel Talavera, Massachusetts Institute of Technology, Cambridge, MA, calitaly@mit.edu
Charuleka Varadharajan, Massachusetts Institute of Technology, Cambridge, MA, charuv@mit.edu
Shaomin Wang, Massachusetts Institute of Technology, Cambridge, MA, smwang@mit.edu
Karim Yehia, Massachusetts Institute of Technology, Cambridge, MA, kyehia@mit.edu
David Zych, Massachusetts Institute of Technology, Cambridge, MA, dzych@mit.edu

Formatted: Font color: Pink (×11)

Abstract *— We have developed a software architecture to support a scalable community of online experiments for use by students at multiple institutions. This work is based on MIT's over five year experience in deploying online laboratories in several engineering disciplines. The iLab architecture provides highly reliable, generic services that are independent of any particular experiment domain, including services for user authentication, authorization, experiment storage, and scheduling. We have been guided by two architectural principles. First, we have striven to free the developers of an online experiment not only from as much code development as possible, but also from user management responsibilities and policy issues. Second, we believe that the architecture should make no assumptions about the platforms used by students, experiment implementers, or university IT support. Clients and servers communicate via web services. We have already completed a reference implementation for "batched experiments", those in which the entire course of an experiment can be specified before execution begins. This implementation has been tested successfully by deploying the MIT Microelectronics WebLab (an online microelectronics device characterization test station) over the Spring 2004 semester in a large undergraduate course at MIT involving over 100 students. In this case the generic Service Broker, implemented in the .NET environment, mediated between a Windows 2000-based Lab Server and a graphic Java client. Students and faculty performed administrative operations using a standard browser over a secure web connection. We are extending the architecture to support interactive experiments and are adding functionality to support searching for attributes in XML-encoded experiment result records. Our goal is for this architecture and our reference implementations to spur the development of new online laboratory experiences and encourage the formation of educational consortia to share the expense and management of online labs. We are already exploring its use to make several MIT labs available to colleagues in Europe, Africa, Asia, and the Middle East.*

Deleted: *campuses*

Deleted: *subject*

Index Terms *— online experiments, web services, scalable architecture, laboratory consortium*

MOTIVATION

The iLab Project has deployed a wide variety of online experiments at M.I.T. since the fall of 2000. The experiments have ranged across a broad spectrum of engineering disciplines including chemical (a heat exchanger [1], polymer crystallization [2]), civil (seismic simulation [3], an instrumented flagpole [4]) and electrical engineering (semiconductor characterization [5-8]). In the first two years, the project explored the nature of online laboratories by encouraging disparate discipline-specific approaches. Our researchers focused on what was perceived as the greatest challenge, giving the student a laboratory experience that was as genuine as possible despite the lack of direct contact with the actual lab equipment. Most of these first-generation online experiments treated the mundane details of student, data, and resource management as an afterthought. Each experiment developed its own approach to validating students, scheduling their sessions with the online lab, and storing or returning the experiment results. Teams working on separate experiments frequently duplicated each other's efforts. It

Deleted: the whole

Deleted: Frequently

Deleted: t

proved surprisingly difficult to provide stable and secure software mechanisms to manage these routine issues. In at least one case, it became necessary to have a graduate teaching assistant camp out beside the lab software server to manually enforce student scheduling because of the lack of an appropriate software mechanism to terminate a student's lab sessions.

More recently, however, we have recognized advantages that arise from a unified approach to online labs. This resulted in a decision over the summer of 2002 to try to build a common software architecture that could suppport experiments from all disciplines without limiting the freedom of the faculty and developers to design and implement the discipline specific portions of their online experiments. We first needed to determine, given the immense variety of laboratory experiments and equipment, whether they share sufficient characteristics so that they can be supported by a common software infrastructure. The fact that hardware vendors now provide low level control of lab devices via industry standard protocols such as GPIB does not address the issue of a high-level software architecture. Packages like National Instruments' *LabView*™ and its associated lower level libraries provide visualization and interface support, but they do not integrate with the types of enterprise scale software that universities are using to manage courses.

We decided the leverage exists in providing general support for the framing and maintenance of a lab session while leaving details of the lab interface and control to domain specific experts or to vendors like National Instruments. That is, we want to distinguish the parts of using an online lab that are specific to a particular piece of lab equipment or to the series of tasks that comprise a particular experiment from the generic tasks that precede, manage, and follow any lab session.

The student will usually need to authenticate himself to the lab software. The student's online identity and affiliation will usually govern the labs that are subsequently made available. For experiments of some duration, the student may need to have previously reserved the online lab. Results need to be stored, analyzed, compared, or printed. The system may encourage collaborative work using standard application sharing and communication tools. The system should allow the student to forward a log of lab activity to a staff member to enlist their help with a problem. These capabilities are generic and transcend the individual experiment. Not all online experiments will require all of them, but most online experiments will require some of them.

We also wanted our unified architecture to address a more subtle problem. The ability to make a lab accessible from the Internet makes it easier to share that lab with colleagues and students at other universities. Provided that network connectivity is sufficient, students can access an online lab from anywhere in the world. The approach of our first-generation labs required registering each student accessing an online experiment on the lab server connected to the lab equipment or on another machine that formed part of the same laboratory network. We soon realized this discouraged lab owners from sharing their equipment. In effect, it penalized faculty members who agreed to share their equipment by obligating them to administer the accounts of users from other universities. If the lab server stored the results of experiment runs, then the staff responsible for the lab server also assumed the responsibility for storing and archiving (or otherwise disposing of) student experiment results at the end of the semester. Different classes often have different policies about how students may team up to execute experiments or share their results afterwards. The owner of a lab server has no desire to manage the student accounts and experiment results of anyone else's students except his or her own. A major goal of the iLab common architecture is to distribute software functionality so that the different participants in the distributed system can most naturally assume responsibility for their own tasks and not hinder or act as proxies for the other participants. Teaching faculty should manage their own students, while lab implementors should only be concerned with implementing and supporting their own labs and experiments.

Formatted: Font: 10 pt, Italic

Formatted: Font: (Default) MS Shell Dlg, 8.5 pt

Deleted: lies

Deleted: specific

Deleted: will

Deleted: a

Deleted: his or her

Deleted: permanently

Deleted: players

Deleted: players

THE iLAB ARCHITECTURAL PROGRAM

From the perspective of online laboratory management, experiments fall into three broad categories:

1. *Batched experiments* are those in which the entire course of the experiment can be specified before the experiment begins. MIT's Microelectronics WebLab [5-8] provides an example. Through WebLab students can characterize a variety of semiconductor devices by preparing a test protocol. This is accomplished by using an interactive editor before the semiconductor characterization executes.
2. *Interactive experiments* are those in which the user monitors and can control one or more aspects of the experiment during its execution. MIT's online Heat Exchanger [1] provides an example. Students can dynamically change the input to heating elements and the action of pumps controlling fluid circulation in the heat exchanger while watching instruments report the changing temperatures.
3. *Sensor experiments* are those in which users monitor or analyze real-time data streams without influencing the the phenomena being measured. MIT's online photovoltaic station [11] provides a simple example.

Each category of experiment requires a different mix of shared services. Since the user completely specifies a batched experiment before execution of the experiment begins, the user need not be online when the experiment is performed but

Deleted: [refs]

Deleted: [ref]

Deleted: (

Deleted:)

instead can retrieve the results later. This implies that batched experiments should generally be queued for execution in a way that maximizes the efficient use of the lab server rather than scheduled to maximize the convenience of the user.

Since the user can control and alter at least some of the inputs of an interactive experiment, however, he or she must be online when the experiment executes. If an experiment run takes more than a few minutes, students and faculty will normally demand that experiments be scheduled so that students will not waste excessive time waiting for their turn at the apparatus.

Sensor experiments usually allow no control except for the ability to subscribe to different data streams that may provide different resolutions or transformations of the base data. They also offer the ability to multicast the same data to many users; in contrast, batched and interactive experiments usually allow individual users or small teams to specify their own experiments that execute sequentially. A sensor experiment usually gathers data over a long period of time for statistical analysis or searches for events of interest in a continuous data stream. Such experiments often provide an archive of past data.

The iLab Project at MIT ultimately plans to provide common services for all three types of online experiments. Both because the batched experiment's simpler requirements and because of our close association with a working online batched experiment (MIT's Microelectronics WebLab), we decided to develop this part of the architecture first. We have tested this stable prototype in a large MIT course with over 100 students (see below) and plan an initial open source release of the common batched experiment source code in the Fall of 2004. Design work for the interactive architecture began during the Spring of 2004, and two of us (Northridge and Yehia) implemented an experiment prototype based on an existing online crystallography experiment [9,10]. We expect to develop the prototype into a full interactive shared architecture this coming Fall, deploy it in an MIT class during the Spring of 2005, and release it in the Fall of 2005. We plan to proceed with the development of the sensor shared architecture in 2005-6.

Deleted: (
Deleted:)
Deleted: deployed a
Deleted: architecture

THE ROLE OF WEB SERVICES

In theory, the software framework and the network technologies we use to build our architecture are implementation issues. With the rise of middleware and distributed application frameworks, however, one's choice of a distributed computing strategy can affect the whole design of a project.

There are design requirements in the iLab Project that immediately favor the use of web services. Students at one institution must be able to use a lab housed at a second institution. This requires an architecture that supports both lab-side services (e.g., the online lab itself) and client or student-side services (authentication and authorization, class management, student data storage for experiment specifications and results as well as user preferences). The lab side services may need to run on a different hardware and software platform than the client-side student software. The lab-side institution may enforce different networking policies (e.g., firewalls, directory and email services) than the client-side institution. The transparency of web services makes this technology an obvious choice to integrate our distributed application framework.

Very often existing labs have a large preexisting code-base to manage the lab equipment or to display results and control the lab equipment from one or more client machines. In the case of Microelectronics WebLab, students had already used a first-generation version of the online lab implemented via a Java applet (the Java client used by the student) and a Windows 2000 Server containing an Agilent GPIB interface board to control the semiconductor analyzer and switching matrix. The loose coupling of web services makes it easier to reuse such legacy code as the basis of a second-generation implementation based on the iLab Shared Architecture. Web services should also make it easier to incorporate vendor supplied modules in the overall architecture.

When we turn to consider the future course of the project, the case for web services only becomes stronger. We have already mentioned that we expect the use of Internet accessible labs will foster increased cooperation between educational institutions. Some schools and colleges may be much more interested in becoming customers of such labs than in offering online labs themselves. They will need a means of discovering what online labs are available in their fields of interest and of verifying that those labs are compatible both with their campus networking and software installed base as well as the pedagogic goals of their courses. Web Services WSDL (Web Services Description Language, http://www.w3c.org/2002-/ws/desc/) and UDDI (Universal Description, Discovery, and Integration, http://www.uddi.org/) provide a framework in which to conduct this discovery process. To carry this one step further, one can imagine WSDL-based negotiation that will match an Internet accessible lab with high end visualization and data analysis tools that are licensed by the client-side institution.

For all the reasons above, we have based our shared software infrastructure for Internet accessible labs on web services. But it is also important to recognize that web services will not solve all the networking requirements of Internet accessible labs. Consider an online sensor lab that wants to multicast high bandwidth sensor data to subscribed clients. While web services may aid the subscription process, they don't provide any support for streaming the multicast data to the students' clients.

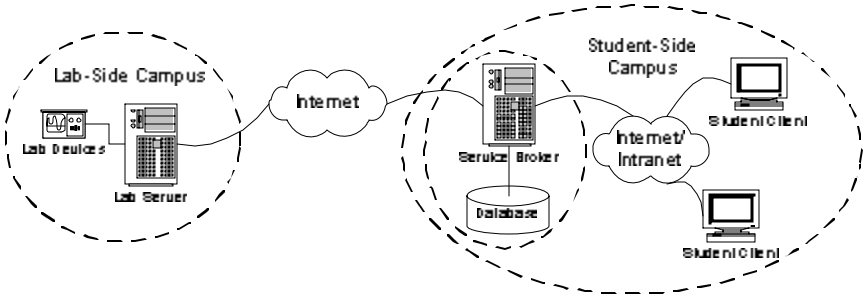
Deleted: on
Deleted: campus
Deleted: on
Deleted: campus
Deleted: campus
Deleted: campus
Deleted: National Instruments
Deleted: stage and
Deleted: [ref]
Deleted: area
Deleted: [ref]
Deleted: [ref]
Deleted: campus

THE BATCHED EXPERIMENT ARCHITECTURE

Our goal in the first phase of developing a shared architecture has been to support batched experiments that cross institution boundaries. We have designed an architecture that in some ways resembles the typical three-tier web business architecture.

1. The first tier is the student's **Client Application** that either runs as an applet or as a downloaded application on the student's workstation.
2. The middle tier, called the **Service Broker**, provides the shared common services. It is backed by a standard relational database such as SQL Server™ or Oracle™. The student's client communicates solely with the Service Broker, which forwards experiment specifications to the final third tier. Unlike the standard three-tier web architecture in which the middle tier resides on the business rather than the client side of the network, we expect the Service Broker will normally reside on a server at the student's institution. If a university is willing to provide accounts for users from other institutions, however, the architecture allows the Service Broker to run on a separate campus from the client; in fact, it can be collocated with the lab itself.
3. This third tier is the **Lab Server** itself, which executes the specified experiments and notifies the Service Broker when the results are ready to be retrieved.

FIGURE. 1
The Topology of the Batched Experiment Architecture.



In this scheme, the student client and the lab server represent the domain- or lab-dependent software modules. We intend the Service Broker to be completely generic code. That is, we expect the operators of the Service Broker to be able to use a standard web browser to configure a fresh version of the Service Broker straight from the software distribution and to register and interoperate with any Lab Server that implements the appropriate Lab Server interface expressed in terms of web service SOAP calls defined in WSDL. The student's client is once again a domain- or lab-dependent piece of code that must understand the protocol for specifying a particular experiment's parameters. A student logs on and manages his or her account using a standard web browser. Once the student chooses the experiment that he or she wishes to execute, the client is launched and communicates with the Service Broker using a second web service interface.

The Lab Server knows nothing about the students using the system, and it only stores experiment specifications and results temporarily. The Service Broker authenticates students, checks on their authorization to contact a particular Lab Server, accepts an experiment specification from the student's client, and waits to retrieve the result once the experiment completes. The experiment specification and results are stored on the Service Broker, which also maintains the link between a student and his experiments. Thus all the resources consumed by a student, except for the runtime resources required to execute the experiment, can be drawn from a Service Broker usually located at the student's institution.

There must be a degree of trust between the Lab Server and the Service Broker, first and foremost because the Service Broker authenticates and vouches for student users. The Service Broker also indicates the student's level of access to the Lab Server by forwarding a string key known as the *effective group* when it submits an experiment specification. The Lab Server does not know on which student's behalf it is executing an experiment. It only knows the requesting Service Broker and the effective group associated with the request. This allows lab suppliers to grant different levels of access to different effective groups on multiple Service Brokers, but it delegates to the Service Brokers all decisions about which students or staff can request experiment execution under the various effective groups.

Deleted: campus
Deleted: c
Deleted: a
Formatted: Space Before: 0 pt, Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0.25" + Tab after: 0.5" + Indent at: 0.5", Don't adjust space between Latin and Asian text, Don't adjust space between Asian text and numbers
Formatted: Font: 10 pt
Deleted:
Formatted: Font: 10 pt
Deleted: on
Deleted: campus
Deleted: on other campuses
Deleted: from
Deleted: ¶
Formatted: Font: (Default) MS Shell Dlg, 8.5 pt
Formatted: Space Before: 0 pt, Don't adjust space between Latin and Asian text, Don't adjust space between Asian text and numbers
Formatted: Font: MS Shell Dlg, 8.5 pt
Deleted: staff
Deleted: cooperate
Deleted: W
Deleted: S
Deleted: on
Deleted: campus

Conversely, the Service Broker knows nothing about the domain dependent nature of the experiments. It forwards an opaque object from the Lab Server to the student's client describing the current lab configuration. When the student submits an experiment specification, it is forwarded to the Lab Server as another opaque object, and the results are returned as a third one. The only part of an experiment that the Service Broker understands is a metadata description of the experiment that can be used to search for and retrieve old experiments. This metadata is implemented as a class that contains fields common to all experiments, e.g., the Lab Server ID, the effective group, etc. It also includes an XML-based extension mechanism that will allow experiment descriptions to be expressed in domain specific XML schemas or DTDs. We assume that the Service Broker at one institution may give its students access to Lab Servers from multiple institutions, and a Lab Server may receive experiment specifications from Service Brokers at multiple institutions.

In the batched experiment architecture the student's workstation never contacts the Lab Server directly. We can maintain this strict discipline because the batched experiment requires so little communication between the client and the Lab Server. Conceptually, the execution of an experiment requires a single round trip over the network although the actual web service protocol is more complicated. This simplicity obviously does not extend to the cases of a streaming sensor experiment or an interactive experiment in which there are strong arguments for having the Service Broker authenticate the student's client but then allow the client to connect to the Lab Server directly.

Outline of a Student Batched Experiment Session

The following walk through of a minimal student experiment session illustrates the batched experiment architecture in action. Table I highlights the interactions and web service calls between the components of the system: Service Broker (SB), Lab Server (LS), and student web browser and Lab Client. The Service Broker also offers a wide range of administrative functionality for student account and experiment record management that is not illustrated in this example.

1. The student starts the session by logging in to the Service Broker (SB) using a secure connection (SSL). The SB provides an implementation of a standard name and password authentication scheme, but it can also integrate with a university's enterprise scale authentication system (e.g., Kerberos).
2. The SB responds by displaying a list of the groups in which the student is registered. Groups usually correspond to classes.
3. The student selects one of the available groups (classes) for this session.
4. The SB displays all the available Lab Clients known for this group. A Lab Client usually corresponds to a single experiment.
5. The student selects one of the available Lab Clients.
6. The SB now launches the Lab Client. This marks the transition in the student's session from communicating with the SB using a web browser to view the SB's active server pages to the running of an experiment during which the student communicates using the Lab Client and web services. We have explored two client technologies: (1) a Java applet that is launched by redirecting to a page with an embedded applet tag, and (2) a client implemented as an active server page launched simply by redirecting to the client page.
7. The student edits the description of the experiment to be run using the client. When the experiment specification is complete, the student directs the client to invoke the web service `Submit()` method on the Service Broker. `Submit()` takes a text encoded version of the experiment specification as an argument. The SB is not expected to understand the specification.
8. The SB stores a copy of the experiment specification and forwards the `Submit()` call on to the LS.
9. The LS immediately returns a submission report that includes any error messages resulting from an illegal experiment specification. If the specification is legal, the LS queues the experiment for execution.
10. The SB forwards the submission report back to the client along with an integer experiment ID that all parties now use to identify the experiment.
11. Once the LS executes the experiment, the LS calls the `Notify()` web service on the SB to indicate that the experiment's results are now available.
12. The SB now requests the results from the LS using the `RetrieveResult()` web service.
13. The LS returns the results and any error messages to the SB, which stores but can not interpret the experiment results.
14. The client, at its leisure, can request the cached results from the SB using the client's the `RetrieveResult()` web service. Note the client need not be logged on while the experiment executes.
15. The SB returns the results and any error messages.

TABLE I
OUTLINE OF A STUDENT BATCHED EXPERIMENT SESSION

Deleted: on
Deleted: campus
Deleted: on
Deleted: campuses
Deleted: on
Deleted: campuses
Deleted: W
Deleted: S
Deleted: it
Deleted: players in
Deleted: a
Deleted: implementation
Formatted: Font: (Default) Times New Roman
Deleted:

STUDENT	SERVICE BROKER	LAB SERVER
Using **Web Browser**	*Using* **Web Application**	
1. Authenticates over SSL ›		
	2. ‹ Lists student's groups (classes)	
3. Chooses group (class) for session ›		
	4. ‹ Lists available Lab Clients (experiments)	
5. Chooses Lab Client (experiment) ›		
	6. ‹ Launches Lab Client	
Using **Lab Client & Web Services (WS)**	*Using* **Web Services (WS)**	*Using* **Web Services (WS)**
7. Calls WS › `Submit (experimentSpecification)`		
	8. Calls WS› `Submit (experimentSpecification)`	
		9. ‹ Returns `SubmissionReport`
	10. ‹ Returns `ClientSubmissionReport` including `experimentID`	
		11. Executes experiment and ‹ Calls WS `Notify(experimentID)`
	12. Calls WS › `RetrieveResult(experimentID)`	
		13. ‹ Returns `ResultReport`
14. Calls WS › `RetrieveResult(experimentID)`		
	15. ‹ Returns `ResultReport`	

FIELD TRIAL OF THE SHARED ARCHITECTURE

The iLab Shared Architecture was test deployed in the Spring of 2004. For this first field trial, we decided to bring online a redesigned version of the MIT Microelectronics WebLab (WebLab 6.0). The Microelectronics WebLab allows students to characterize the current-voltage behavior of transistors and other microelectronic devices. Since it was first deployed in the Fall of 1998, it has been used by over 2000 students for graded assignments from three continents [5-8].

Deleted: selected

Deleted: characteristics

Deleted: nl02, icee 02 and 03

Deploying the Microelectronics WebLab through the new Shared Architecture required the construction of two new web-services interfaces for Java client and the lab server to interface to the Service Broker. In our first generation WebLab (v. 1.0-5.0), the software did not allow for a clean split of user management from experiment execution functions. As a result, for this trial we constructed a new lab server from scratch. Similarly, we completely redesigned the Java client application. Its look and basic ergonomics are the same as that of WebLab 5.0 [8], but its internal architecture is now far more modular. A screen shot of the Java client is shown in Figure 2. The details of the design of WebLab's Java client and the Lab Server will be described in more detail in a separate publication.

Deleted: the Service Broker to interface to the

Deleted: system partition

Deleted: was

Deleted: from scratch

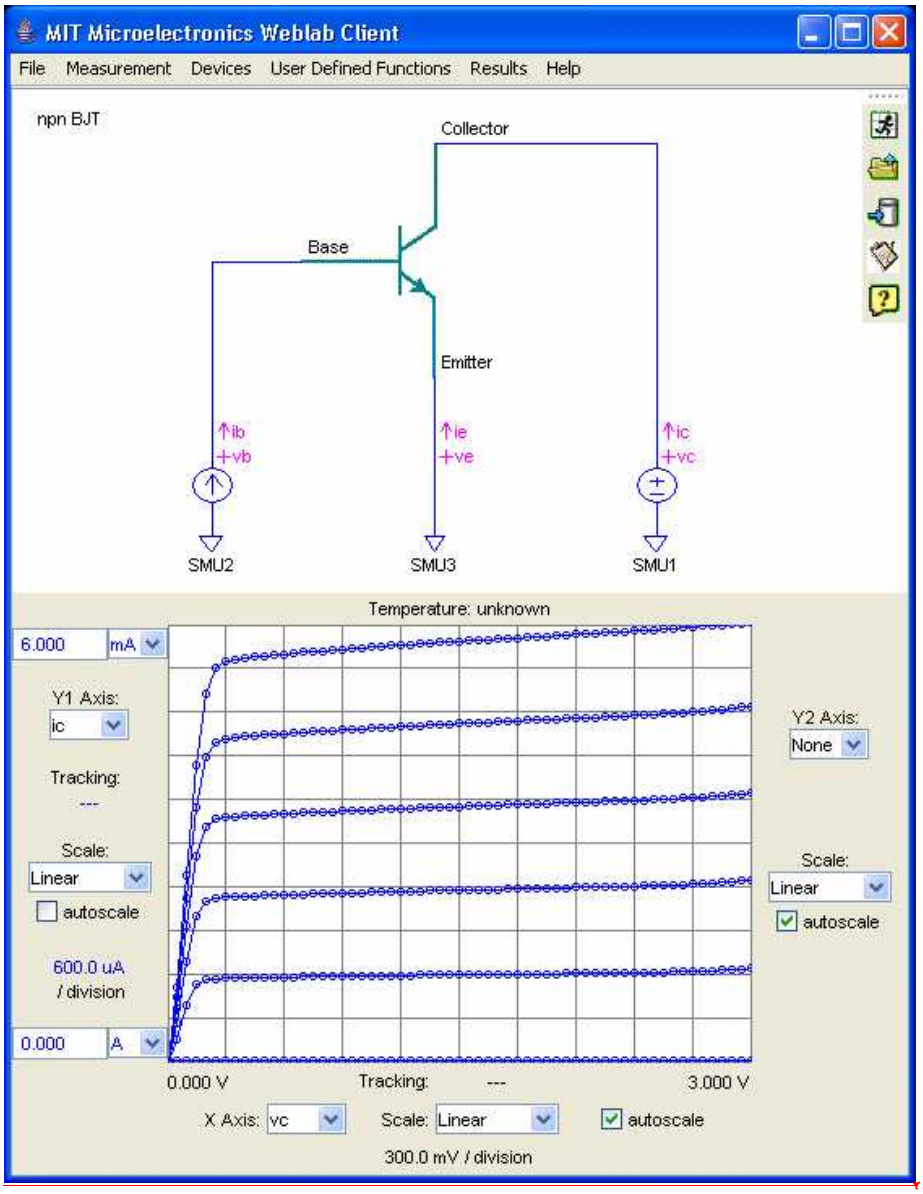
Deleted: icee 2003

Deleted: It also interfaces with the Service Broker through web services.

Deleted: . X

Deleted: separately

FIGURE. 2
Screenshot of the WebLab 6.0 Java Client showing an experiment in which the output characteristics of a bipolar transistor are obtained.



The trial WebLab 6.0 and the iLab Shared Architecture was conducted in the Spring of 2004 in a junior-level subject in the Department of Electrical Engineering and Computer Science at MIT with about 100 students. Two assignments were given. These were of a similar character to those required in earlier editions of this class using previous versions of WebLab. In the

Formatted: Page break before

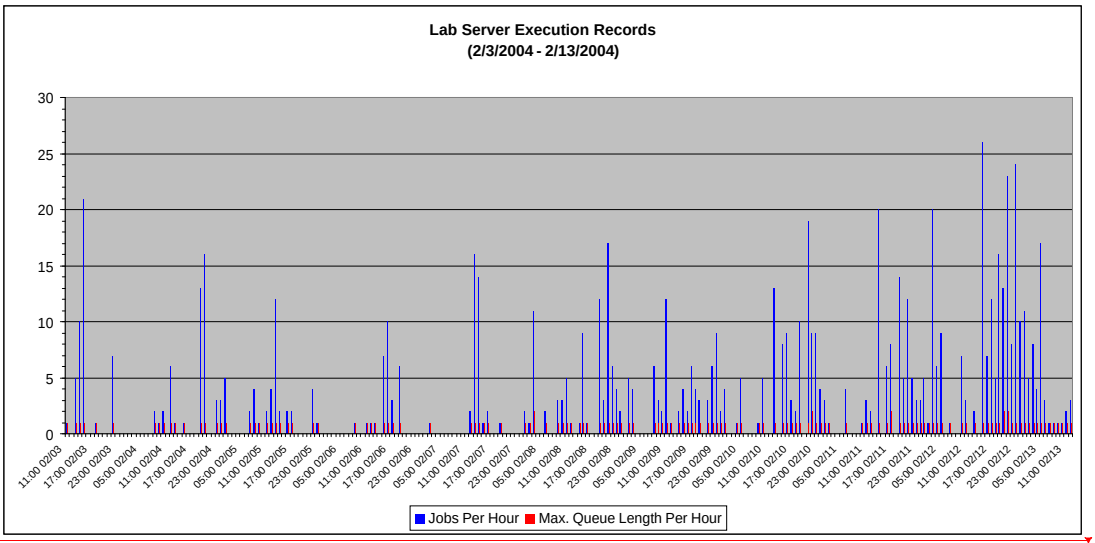
Deleted:

Deleted: was deployed through

first assignment, students were asked to characterize and model a pn diode. In the second assignment, students were asked to characterize an npn bipolar transistor and an n-channel MOSFET and then asked to design a simple common-collector amplifier. Both were week-long assignments.

During these assignments, the system worked nearly flawlessly. A histogram of usage for the first assignment is shown in Figure 3. As seen, the system handled all requests expeditiously. The queue of the system never held more than two jobs.

FIGURE. 3
Histogram of Experiment Queueing and Execution During the WebLab 6.0 Trial, Spring 2004.



Our plan is for this new system to become the default version of WebLab in the Fall of 2004. For the Fall academic semester, we have accepted commitments for the system to be used by over 1000 students from four continents.

THE iLAB COMMUNITY

The ultimate goal of the iLab Project is to provide a much richer set of experiment resources for students at MIT and elsewhere. The iLab architecture should encourage educational institutions to share both laboratory facilities and carefully designed lab exercises based on those facilities. While such sharing is crucial to the growth of a community of iLab providers and users, the members of this community need not all make identical contributions. The experience of MIT's Microelectronics WebLab has already shown that colleagues at other universities can contribute to the educational value of the lab by making their faculties' lab assignments publicly available. During the Fall 2004 semester, we have accepted commitments for the system to be used by over 1000 students from four continents. A second batched experiment on feedback systems has come online at MIT this summer [12] and will be used in a course this coming fall. Over the course of the next academic year we hope to see new experiments come on line at universities in Sweden, Lebanon, and Taiwan.

At the same time, we plan to make a wider community aware of the iLab architecture through a number of initiatives. In cooperation with the Open Courseware Initiative (OCW) at MIT, we intend to set up a special Service Broker at MIT that will offer public access to a version of the Microelectronics WebLab. The initial batched experiment Service Broker source code will be released in stages under an open source license this Fall and Winter. Finally, we intend to hold a workshop at MIT in January, 2005, for potential iLab developers.

The iLab initiative will only grow if colleagues comment on the architecture and contribute to the set of online labs and associated educational materials. We welcome the involvement of contributors at every level. The success of the iLab

Deleted: y
Deleted: . Y
Formatted: No page break before
Deleted:
Deleted: [more from the rest of the team here with more postmortem?]¶
Deleted: s
Deleted: is to
Deleted: to use
Deleted: we expect the MIT Microelectronics WebLab to be used by over 1000 students from institutions in countries ranging from Sweden to Taiwan

concept will ultimately be measured by the degree to which it fosters cooperation and sharing between institutions, experimenters, faculty, and students.

ACKNOWLEDGEMENT

This project is funded by Microsoft through iCampus, the MIT-Microsoft Alliance. The instruments used in WebLab were donated by Agilent Technologies.

REFERENCES

[1] http://heatex.mit.edu.

[2] http://web.mit.edu/rutledgegroup/projects/onlinelab.html.

[3] http://flagpole.mit.edu:8000/shaketable/.

[4] Amaratunga, K., and R. Sudarshan, "A Virtual Laboratory for Real-Time Monitoring of Civil Engineering Infrastructure", ICEE, Manchester (UK), 2002.

[5] del Alamo, J. A., L. Brooks, C. McLean, J. Hardison, G. Mishuris, V. Chang, and L. Hui, "The MIT Microelectronics WebLab: a Web-Enabled Remote Laboratory for Microelectronic Device Characterization", World Congress on Networked Learning in a Global Environment, Berlin (Germany), 2002.

[6] del Alamo, J. A., L. Brooks, C. McLean, J. Hardison, G. Mishuris, V. Chang, and L. Hui, "Educational Experiments with an Online Microelectronics Laboratory", ICEE, Manchester (UK), 2002.

[7] del Alamo, J. A., L. Brooks, C. McLean, J. Hardison, G. Mishuris, V. Chang, and L. Hui, "MIT Microelectronics WebLab", chapter in T. Fjeldly and M. Shur, Eds., *Lab on the Web - Running Real Electronics Experiments via the Internet*, Wiley-IEEE, 2003, pp. 49-87.

[8] del Alamo, V. Chang, J. Hardison, D. Zych, and L. Hui, " An Online Microelectronics Device Characterization Laboratory with a Circuit-like User Interface", ICEE, Valencia (Spain), 2003.

[9] Northridge, Jedidiah, "A Federated Time Distribution System for Online Laboratories", MIT Master of Science thesis, May, 2004.

[10] Yehia, Karim, "The iLab Service Broker: a Software Infrastructure Providing Common Services in Support of Internet Accessible Laboratories", MIT Master of Science thesis, May, 2004.

[11] http://pvbase.mit.edu/cgi-bin/index.py.

[12] http://web.mit.edu/6.302/www/weblab.

Formatted: Font color: Pink
Formatted: References, Justified
Formatted: Font color: Pink
Formatted: Font color: Pink
Formatted: Font: Not Italic
Formatted: Space After: 6 pt
Formatted: Font: Not Italic
Formatted: References, Justified
Formatted: Font: (Default) Times New Roman
Formatted: Font: Italic
Formatted: Font: (Default) Times New Roman
Formatted: Font color: Pink