Deploying Interactive Remote Labs Using the iLab Shared Architecture

James L. Hardison, Kimberly DeLong, Philip H. Bailey, and V. Judson Harward
Massachusetts Institute of Technology, hardison@mit.edu, kirky@mit.edu, pbailey@mit.edu, jud@mit.edu

Abstract - **The MIT iLab Project has developed a distributed service infrastructure and software toolkit to support a scalable community of online laboratory experiments. The iLab Shared Architecture provides a framework for the development and deployment of remote laboratories using a three-tiered model based on web services consisting of lab clients, service broker middleware, and lab servers. This simplifies the development of remote labs by providing reusable components for common lab administration functions. The initial focus of the iLab Project was on batched labs, which require no interactive control. Following the project's success in supporting these labs, it has expanded its efforts to include those requiring interactive control. Interactive labs require that the user have active control of lab instruments during the course of an experiment and can generate a large amount of data. In order to accommodate these requirements, the iLab Shared Architecture has been extended with a highly configurable lab resource scheduling service, a robust data storage system and support for high bandwidth communication between the lab client and server. By integrating these services into the iLab Shared Architecture, a more diverse set of educationally valuable labs can now be easily deployed online and shared around the world.**

Index Terms – Educational Technology, Engineering Education, Internet, Laboratories

INTRODUCTION & BACKGROUND

Since 1998, effort has been put towards the development of remotely accessible online laboratories, or iLabs, at MIT. Initially, these were ad hoc efforts by individual faculty who were dissatisfied with the laboratory experiences available to their students [1]-[4]. One such remote lab, the MIT Microelectronics WebLab, was born out of an introductory microelectronics course that, traditionally, was focused on the study of theoretical device models. Adding a laboratory component to this class would be educationally valuable, but a hands-on lab would not be feasible. Lab instruments were costly, suitable space was limited and the logistical requirements for such a lab were prohibitive [5]. Faced with this situation, Prof. Jesús A. del Alamo initiated the development of the Microelectronics WebLab, which allowed students to perform laboratory experiments at any time via the Internet using a single semiconductor parameter analyzer in a research lab.

The Microelectronics WebLab and other similar efforts shared goals and were successful, but grew independently. This resulted in a number of different approaches to the task of providing access to an online lab as well as some duplicate efforts by the individual groups. The MIT iLab Project was formed with the goal of defining a standard approach to the development of online labs and providing tools to make such development simpler for those wanting to create new labs. To this end, the iLab Project developed a distributed service infrastructure termed the iLab Shared Architecture [6].

In broad terms, the iLab Shared Architecture (ISA) divides an online lab into three distinct parts: the lab client, the Service Broker and the lab server. The lab client is the user's interface to the iLab while the lab server connects to the lab hardware and manages the execution of user submitted experiments. The ISA specifies that lab clients and lab servers contain lab-specific functionality. The Service Broker is responsible for providing functionality that is generic and useful to all iLabs. Services such as user authentication/authorization and data storage are built into this middleware layer. The ISA provides a framework for the deployment of iLabs in a distributed fashion using web services. This allows online labs developed on the ISA to be made available to users worldwide using standard network protocols.

In addition to aiding the development of remote labs, the distributed nature of iLabs encourages the sharing of these resources. By placing a Service Broker at an institution, its users can access iLabs without creating an administrative burden to the host of the lab. University A can deploy an iLab and manage its own students using its own Service Broker. Meanwhile, it can share this iLab with University B who, with the use of its own Service Broker, can manage its own students. In this way, there is little or no cost to sharing an iLab beyond the instrument's time and any consumable resources.

Initial work on the ISA centered on support for what were termed "batched" labs. These are labs where experiments are completely specified prior to submission and run without intervention. Batched iLabs are deployed with lab clients, Service Brokers and lab servers that communicate over the Internet using web services. In this model, detailed in Figure 1, lab clients and lab servers communicate with each other exclusively through the

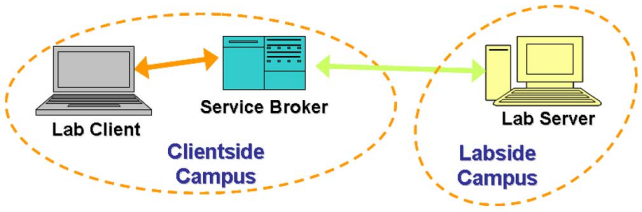

FIGURE 1
TOPOLOGY OF A BATCHED EXPERIMENT BASED ON THE ILAB SHARED ARCHITECTURE.

iLab Service Broker. This ensures that communications between these systems are standardized. Any batched type remote lab that is iLab compliant will be compatible with, and sharable through, any Batched Service Broker in the world.

In 2004, the MIT Microelectronics WebLab was redeployed on the ISA and has been successfully used in for-credit courses both at MIT and 18 other institutions worldwide [7]-[8]. Since then, additional batched iLabs have been developed at MIT, the University of Queensland, Australia and at Obafemi Awolowo University, Nigeria [8].

Following this work, the iLab Project focused on supporting "interactive" labs. These are experiments that involve some manner of real-time control or observation. Many traditional labs fit this model and adding support for them in the ISA would enable the deployment of a much broader set of iLabs.

Interactive experiments are fundamentally different from their batched counterparts. Primarily, interactive experiments require control of lab hardware while the user sets parameters and observes results. This is in contrast to the batched model where experiments are queued and run when the lab hardware is available. An interactive experiment, then, must commit the lab hardware to a single user for the duration of their session - typically 20 minutes to an hour – and may require scheduling.

Another main difference between interactive and batched labs involves the role of the Service Broker. Interactive experiments require real-time control and, potentially, much greater bandwidth between the lab client and the lab server. Because of this, the batched notion of a Service Broker that uses web services to route all communications between the lab client and lab server will not work effectively in an interactive iLab.

In order to support the deployment of interactive experiments, the iLab Project would have to revise and extend the iLab Shared Architecture.

SUPPORTING INTERACTIVE LABS IN THE ILAB ARCHITECTURE

From the user's perspective, an interactive iLab provides a higher degree of control than its batched counterpart. As in the batched case, the student logs on to their Service Broker to gain access to the lab. Since an interactive lab grants full control of the lab hardware to the user for a relatively long period of time, the user must have previously scheduled time to use the experiment. Having done this, the user is able to launch the lab client, which presents the interface to the interactive iLab. At this point, the user has full control of the lab hardware. They can submit experiment parameters and commands as well as observe the resulting behavior of the experiment. The user can adjust parameters or submit new commands in real-time. All interaction between the user and the lab can be saved to form a definitive record of the experiment session.

The iLab Shared Architecture dictates that, for batched labs, the lab client and the lab server communicate through an iLab Service Broker using web services. This ensures that labs deployed on the iLab standard remain easily accessible across the Internet and that users are properly authenticated before being granted access to a lab. This also allows the user's Service Broker to construct a complete record of their experiments. As all lab communications – lab configurations, experiment specification and data – pass through a Service Broker, it is trivial for the Broker to also record that information. These features are valuable to the broad set of remote labs, not just batched ones.

Developers of interactive labs need the same data storage capability as well as access to the other generic services available to batched labs on the ISA. However, the nature of an interactive experiment is such that the methodologies used for batched labs do not apply. Interactive experiments require real-time or close to real-time control of lab hardware. Further, such active control must be achievable for a variety of observational modes. Most experiment information in batched labs is transmitted as text-based messages. However, an interactive lab developer may want students to observe, for instance, video of a sample through a microscope. Developers of interactive labs need the freedom to utilize those communication protocols that best fit their needs while still having access to the features of the iLab Shared Architecture.

The interactive iLab topology is detailed in Figure 2. In addition to the lab client, Service Broker and lab server, stand alone web services are added to manage experiment storage and lab scheduling. As in the batched model, a user begins his or her session on an interactive lab by logging on to the Service Broker at their university and authenticating themselves to the system. The student must first schedule a time to use the lab. At the scheduled time, the student logs in and is able to launch a lab client. This client interacts directly with the lab server and, once a lab session begins, the Service Broker steps out of the picture. Using "tickets", the Service Broker can authorize and vouch for the user to the lab server in a way that can be preserved on the lab server for the duration of the scheduled experiment session.

An interactive lab must support the ISA web service interfaces in order to take advantage of these added features. The use of web services for these generic services ensures portability of the ISA across different networks and platforms. The ISA, however, makes few assumptions about the nature of direct communication between the lab

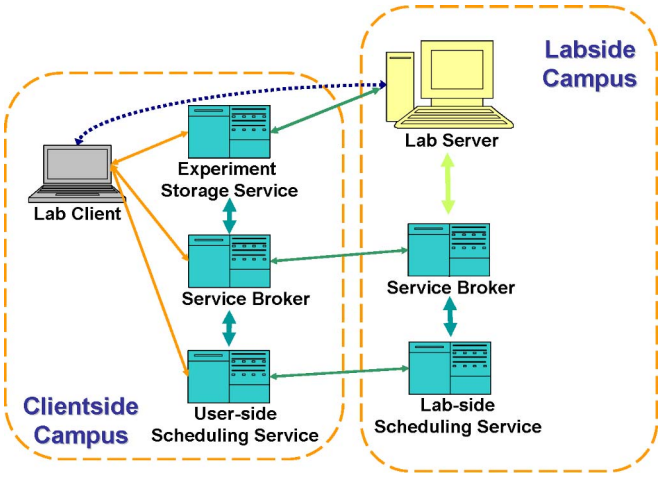

FIGURE 2
TOPOLOGY OF AN INTERACTIVE BATCHED EXPERIMENT BASED ON THE ILAB SHARED ARCHITECTURE.

client and lab server. Any technology can be employed for lab client to lab server communication so long as it supports the implementation of control mechanisms that the ISA can use to enforce lab management policy (i.e. closing the session of a user who runs over into another user's scheduled time). This flexibility enables both higher performance communication and the use of the ISA's generic interactive services.

DESCRIPTION OF THE INTERACTIVE SERVICES

In the iLab Shared Architecture, the Service Broker is responsible for user and group management, authentication and authorization as well as experiment storage and retrieval. However, the interactive model requires that these features be implemented in a new way. First, in an interactive iLab, lab clients and servers communicate directly, so the Service Broker cannot construct a record of an experiment merely by "listening in" on the traffic. Therefore, the lab server and the lab client must be involved in the storage process. Second, interactive experiments typically require that students have exclusive use of lab hardware for blocks of time, so resource management and scheduling is a concern. Third, an interactive experiment is not a set of well-defined, individually verifiable transactions that pass through the Service Broker, as in the batched model. This creates a need for a powerful yet flexible authentication and authorization mechanism to enable secure access to remote lab resources. To support these functions for interactive labs, three new services have been added to the ISA: the Experiment Storage Service, Scheduling Services and Ticketing.

I. The Experiment Storage Service

In batched iLabs, the burden for storing the definitive record of a student's experiment rests with the Service Broker. Generally speaking, this is a sensible decision as it is likely to be close to the student, in terms of network topology. If the Service Broker is located on the student's campus, its managers will likely be aware of the student's academic schedule when setting policy on the longevity of data. For batched iLabs, all experiment communications can be recorded by the Service Broker with little effort since all of this information passes through the Service Broker. This is not the case for interactive experiments.

In an interactive experiment, the Service Broker sets up a relationship between a student using the lab client and the lab server. After this relationship is established, the client and the lab server communicate with each other directly. The Service Broker is no longer privy to experiment parameters, commands or results. This suggests that the storage of experiment data should be handled by a separate service that can be accessed by the Service Broker, lab client or lab server as needed.

In the interactive model, this functionality is provided by the Experiment Storage Service (ESS). The ESS is a stand-alone web service that allows Service Brokers, lab servers and lab clients to store experiment data. As an independent system, a single ESS can potentially be used by many Brokers and their associated labs.

Records on the ESS consist only of experiment data. This includes binary data (images, video or audio) and XML based text/numeric data. Administrative data describing an experiment, such as the student it belongs to, is stored on the Service Broker responsible for that experiment. As such, individual Service Brokers are still able to set customized policy regarding the longevity of experiment records. Students must also use their Service Broker, either directly or in conjunction with an appropriate lab client or analysis program, to access data from a completed experiment.

II. The Scheduling Services

Batched experiments require lab hardware to be committed to a given student only after the experiment is submitted but not beyond the point where data is generated. In many of the batched labs implemented thus far, a single experiment takes 10 to 100 seconds of instrument time. Interactive experiments require the use of hardware while the student controls and observes the experiment, extending the length of a single control session to the tens of minutes or longer. Thus, the batched lab strategy of mitigating high usage loads with a lab-based queuing system does not work in the interactive case.

When one thinks of a traditional, hands-on lab, access is typically managed using a lab schedule. There are hours when the lab is available and students can sign up for blocks of time where they alone can use the equipment. This is the model the ISA applies to managing the use of interactive experiments. Scheduling Services in the ISA consists of two separate, web services-based systems, the User-side Scheduling Service (USS) and the Lab-side Scheduling Service (LSS).

The Service Broker is a critical agent in the scheduling process as it alone can authenticate a user and determine whether they are authorized to schedule time on a given lab.

Once authorized, the User-side Scheduling Service is used in conjunction with the LSS to allocate lab time to users. Using the USS, a student who wants to schedule time on a given lab must select from a set of available blocks of time published by the LSS. Once a time is selected, the USS stores the reservation for later redemption and the LSS removes it from the list of available time blocks. Additionally, the USS is responsible for notifying students if their reservation must be canceled and for considering course/lab requirements when distributing time blocks. For instance, a course instructor may only want students to use the lab for 20 minutes at a time. Such requirements must be factored in when allocating student time on a lab.

The Lab-side Scheduling Service is responsible for defining scheduling policy for a particular lab. The LSS is designed to run in conjunction with multiple USS's and may schedule multiple lab servers. From the LSS, a lab administrator can set lab specific policy. This includes any instrument warm-up/cool-down requirements as well as the periods of time allocated to groups of students on a given Service Broker. Thus, a LSS defines the broad lab availability for individual USS/Service Brokers. In turn, a given USS/Service Broker will distribute experiment time to students based on lab requirements, instrument availability and instructor policy.

Not only do the Scheduling Services define when a student can use the lab, but they also dictate when that student must relinquish control. In short, if an interactive iLab is to take advantage of Scheduling Services in the ISA, it must be built in such a way that scheduling can be enforced.

III. Ticketing

A relatively simple authentication/authorization model can be employed by batched labs in the ISA since all communication is routed through the Service Broker. In the batched model, the student logs on to their Service Broker using a common two-factor sign on (username and password). At that point, the student is authenticated to the Service Broker web application and is presented with the labs they are authorized to use. Since lab clients communicate through the Broker, the session-based authentication passes easily between the student's web browser and their lab client (typically a web form or a Java applet). Between the Service Broker and a lab server, another two-factor, credential-based authentication system is used to prove a Broker's identity to the lab server. This information is sent with each call to the lab server and can be secured using SSL. In the interactive model, this credential system has been expanded to support communication between Service Brokers and each of the ISA services.

The topology of an interactive lab is both more complex and more variable than its batched counterpart. An interactive lab has the same basic components; lab client, Service Broker and lab server. However, there are also Experiment Storage Services and Scheduling Services that all interact with each other. The Service Broker is still responsible for authenticating users and authorizing use of lab server resources. In the case where the student's Service Broker and the lab server are at different institutions, still more Service Brokers are employed to ensure the proper handling of user credentials. In order for there to be a coherent notion of accountability in the ISA, a single overriding authentication system had to be constructed.

This was implemented in the form of Ticketing. In this model, specific services in the ISA are able to, on their own accord or on behalf of other services, issue tickets that permit access to resources. The validity of a ticket is based on the fact that only the issuing and redeeming agents (i.e. Service Broker and lab server, respectively) access the ticket. A ticket stub, called a "coupon", which is used to reference a collection of tickets, is the authorization item transmitted between services in the ISA. Ticketing is used to provide authentication between Service Brokers, User-side Scheduling Services and Lab-side Scheduling Services.

This system can be illustrated by considering the case of a student logging in and running an experiment. Once the student logs in and is authorized to perform an experiment, tickets permitting experiment execution and data storage are created along with a coupon representing the collection. This coupon is passed to the student's instance of the lab client when it is launched. In order to connect to the lab server and begin the experiment, the lab client sends the coupon to the lab server, which retrieves the execution ticket from the issuing Service Broker. If a valid ticket is returned, the student is authorized for a particular amount of time and the experiment proceeds unhindered. Similarly, when experiment data needs to be recorded, the ticket coupon is passed to the ESS, which redeems the data storage ticket. Each ticket only needs to be redeemed once per session.

As with Scheduling and Experiment Storage, Ticketing is based on web services and requires that the developer of the interactive lab provide support for this service.

DESCRIPTION OF THE INTERACTIVE LAB CLIENT/SERVER MODEL

Outside of the services available by the generic portions of the iLab Shared Architecture (ESS, Scheduling, Ticketing), which require communication via web services, decisions regarding the construction of an interactive experiment are left to the lab developer. This allows developers to use custom, even proprietary, technology both for constructing their lab client and lab server and for managing the experiment based communication between them. Not only does this provide support for specific, potentially high-bandwidth lab experiments but it also enables support for pre-existing lab control software.

There is a good deal of interest in constructing iLabs that use the National Instruments LabVIEW® platform for lab hardware control. This is a common development environment among lab domain specialists – those typically tasked with building iLabs.

Responding to this, the iLab Project has developed a reference implementation called the LabVIEW® Integrated Interactive Lab Server (LVILS). The LVILS provides a standard way for interactive lab developers to interface the generic ISA services to LabVIEW® instrument control software. In the LVILS, as in the interactive model generally, the lab client is developed in close relation with the lab server and corresponding instrumentation. The LVILS furnishes support for accessing the ESS, interacting with LSS and USS implementations, properly handling Ticket-based authentication, as well as interfacing to LabVIEW® applications. Using the LVILS, an experienced lab developer can take a working stand-alone experiment built with LabVIEW® and deploy it as an interactive lab on the ISA in a matter of hours [8].

In the case where LabVIEW® is not the chosen technology for the development of an interactive experiment, the approach is similar to that taken with the LVILS. Communication between the lab client and server must be handled by the chosen technology. In turn, the lab client and server must implement the web service interfaces needed to access the generic functionality of the ISA. The iLab Project expects that, as institutions work to develop interactive experiments on the ISA, further use cases will be identified and implementation examples constructed.

DEPLOYING INTERACTIVE EXPERIMENTS ON THE ILAB SHARED ARCHITECTURE

In June, 2007, the first version of the interactive version of the iLab Service Broker was released to the public [9]. This includes an Interactive Service Broker, Experiment Storage Service, Scheduling Services and Ticketing as well as a sample lab server with support for LabVIEW®. As with all releases of the ISA, this reference implementation is released under an open source license. Following from that, two interactive labs, both employing LabVIEW®, have been developed at MIT for deployment on the ISA.

I. Force on a Dipole iLab

The Force on a Dipole iLab is an electricity and magnetism experiment designed for first year physics students at MIT. This iLab is part of a broader project to augment first year physics courses with both hands-on and remote experiments coupled with visualizations in order to illustrate concepts that can be quite abstract [10].

The experiment consists of a small magnet suspended by a spring between two horizontally mounted coils (a Helmholtz coil). Using a LabVIEW®-based client published as an interactive iLab, students can vary the current in the coils in such a way that the magnet oscillates. The lab hardware and client are shown in Figure 3. Based on their measurements and a few known system parameters, students can then determine the magnetic dipole moment of the magnet and, with the aid of visualizations and video of the lab equipment, develop a sense of the electromagnetic forces at work.

FIGURE 3
LAB INSTRUMENTATION (LEFT) AND LABVIEW® CLIENT INTERFACE (RIGHT) FOR THE FORCE ON A DIPOLE ILAB.

The Force on a Dipole iLab will be used by a select group of physics students during the Spring 2008 term. Based on the performance of the iLab and the feedback received from students and instructors, it is projected that this experiment will be used by MIT's mainstream physics course in the Spring 2009 term (approx. 600 students).

II. Nuclear Reactor iLab

The MIT Nuclear Reactor iLab is an online laboratory that exposes some of the functionality of the MIT Nuclear Research Reactor. The reactor is managed by the MIT Nuclear Reactor Laboratory which conducts and supports research in the areas of nuclear energy, nuclear science, medicine and radiation science and technology for students/researchers at a variety of levels [11].

Similar to the Force on a Dipole iLab, the Nuclear Reactor iLab provides a LabVIEW®-based interface to the lab hardware. This client interface is shown, along with the lab hardware, in Figure 4. This lab consists of a neutron beam port, beam aperture and transmission sample plates, an absorption sample and various measurement instruments. Three distinct types of measurements are available using this iLab [12]. Students can measure the Maxwell Boltzmann Distribution of Thermal Neutrons, the diffraction of a pulsed neutron beam and, finally, the neutron absorption behaviors of certain materials.

As of this writing, the Nuclear Reactor iLab is in the final stages of development and is to be used by students

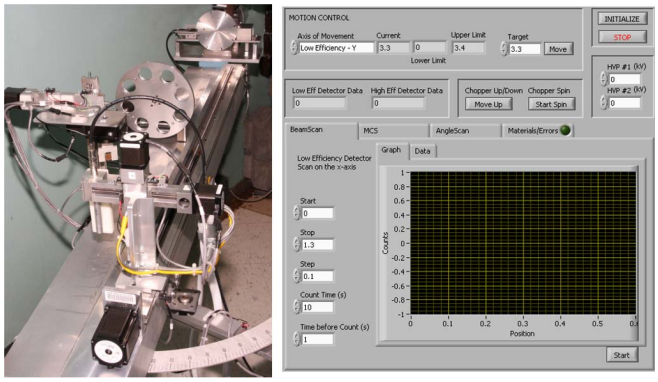

FIGURE 4
LAB INSTRUMENTATION (LEFT) AND LABVIEW® CLIENT INTERFACE (RIGHT) FOR THE NUCLEAR REACTOR ILAB.

during the Spring 2008 term. The initial deployment will be in a course in the MIT Nuclear Engineering department. Further deployments should also involve courses offered by the MIT Physics department and, potentially, local secondary schools.

CONCLUSIONS AND FUTURE WORK

The iLab Shared Architecture provides a flexible software infrastructure for the implementation of Internet-enabled labs. A new version of this architecture has recently been released to support the deployment of interactive experiments. This consists of a set of new services and functionality that are available with the iLab Service Broker, a general model for the development of interactive experiments and software to enable rapid publishing of LabVIEW®-based experiments online. This has enabled the development of a set of interactive iLabs that are quickly being adopted by educators. These iLabs broaden the set of remote experiments available on the iLab Shared Architecture. On a longer time horizon, these experiments will serve to further demonstrate the value of iLabs as a means of providing a rich set of laboratory experiences to engineering students.

Moving on from this work, the iLab Project intends to further refine the interactive services provided by the iLab Shared Architecture. This should lead to a single Service Broker that will support both batched and interactive experiments. This will reduce the barrier to adoption of both interactive experiments and iLabs generally as fewer software systems will need to be deployed. The project also intends to continue to foster the development of batched and interactive experiments at MIT and other institutions.

ACKNOWLEDGMENT

The authors would like to thank Imad Jabbour, Tingting Mao, Loai Naamani, Jedediah Northridge and Rabih Zbib for their efforts in the specification and development of the interactive extensions to the iLab Shared Architecture as well as Jesús A. del Alamo and Steven Lerman for their continued guidance.

This work has been supported in part by the Microsoft Corporation through iCampus (the MIT-Microsoft Alliance), by the Carnegie Corporation of New York, by the National Science Foundation under award #0702735, by the Singapore-MIT Alliance, by the Singapore-MIT Alliance for Research and Technology, and by MIT Alumni Funds (Classes of ’51, ’55, ’60, and ’72) as well as by equipment donations from National Instruments, Agilent Technologies, AMD, Hewlett-Packard, and Intel.

REFERENCES

[1] del Alamo, J. A., Brooks, L., McLean, C., Hardison, J., Mishuris G., *et al.*, “The MIT Microelectronics WebLab: a Web-Enabled Remote Laboratory for Microelectronics Device Characterization”, 2002 World Congress on Networked Learning in a Global Environment, Berlin (Germany), May 2002.

[2] Colton, C. K., Knight, M. Q., Khan, R., West, R., "A Web-Accessible Heat Exchanger Experiment", *INNOVATIONS 2004: World Innovations in Engineering Education and Research*, Win Aung, Robert Altenkirch, Tomas Cermak, Robin W. King, and Luis Manuel Sanchez Ruiz. Arlington, VA: iNEER, 2004, pp. 93-106.

[3] Talavera, D., "On-Line Laboratory for Remote Polymer Crystallization Experiments Using Optical Microscopy", MIT M.Eng. Thesis, 2003.

[4] Amaratunga, K., Sudarshan, R., “A Virtual Laboratory for Real-Time Monitoring of Civil Engineering Infrastructure”, presented at the International Conference on Engineering Education 2002, Manchester (UK), August 18-22, 2002.

[5] del Alamo, J. A., Chang, V., Brooks, L., McClean, C., Hardison, J., *et al.*, “MIT Microelectronics WebLab”, *Lab on the Web*, Tor. A. Fjeldly and Michael S. Shur, Hoboken, N. J.: IEEE Press and John Wiley & Sons, 2003, ch. 2, pp. 49-87.

[6] Harward, J., del Alamo, J. A., Choudary, V. S., DeLong, K., Hardison, J. L., *et al.*, “iLabs: A Scalable Architecture for Sharing Online Laboratories”, presented at the International Conference on Engineering Education 2004, Gainesville, Florida, October 16-21, 2004.

[7] Hardison, J. L., Zych, D., del Alamo, J. A., Harward, V. J., Lerman, S. R., *et al.*, “The Microelectronics WebLab 6.0 – An Implementation Using Web Services and the iLab Shared Architecture”, presented at the International Conference on Engineering Education and Research 2005, Tainan, Taiwan, March 1-5, 2005.

[8] Harward, V. J., del Alamo, J. A., Lerman, S. R., Bailey, P., Carpenter, J, *et al.*, "The iLab Shared Architecture: A Web Services Infrastructure to Build Communities of Internet Accessible Laboratories", *Proceedings of the IEEE* Vol. 96, No. 6, June 2008

[9] “MIT iCampus: iLabs”, http://icampus.mit.edu/ilabs/.

[10] Dori, Y. J., Belcher, J., “How Does Technology-Enabled Active Learning Affect Undergraduate Students’ Understanding of Electromagnetism Concepts?”, *Journal of the Learning Sciences* Vol. 14, No. 2, April 2005, pp 243-279.

[11] “MIT Nuclear Reactor Laboratory”, http://web.mit.edu/nrl/www.

[12] “MIT Reactor iLab”, http://norbert.mit.edu/reactor/index.html.

AUTHOR INFORMATION

James L. Hardison, Research Engineer, Center for Educational Computing Initiatives, Massachusetts Institute of Technology, hardison@mit.edu.

Kimberly DeLong, System Manager/Senior Programmer, Center for Educational Computing Initiatives, Massachusetts Institute of Technology, kirky@mit.edu.

Philip H. Bailey, Senior Project Manager, Center for Educational Computing Initiatives, Massachusetts Institute of Technology, pbailey@mit.edu.

V. Judson Harward, Principal Research Scientist and Associate Director, Center for Educational Computing Initiatives, Massachusetts Institute of Technology, jud@mit.edu.