In Proceedings of the *2007 International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS),* Las Vegas, Nevada, June 25-28.

Collaboratory: An Open Source Teaching and Learning Facility for Computer Science and Engineering Education

Jeff Wright, Stefano Carpin, Alberto Cerpa, German Gavilan
Marcelo Kallmann, Cameron Laird, Kyler Laird, Shawn Newsam and David Noelle
{*jwright,scarpin,acerpa,ggavilan,mkallmann,klaird,snewsam,dnoelle*}*@ucmerced.edu*
Computer Science and Engineering, School of Engineering,
University of California, Merced (UCM)
5200 North Lake Road, Merced, CA 95344

Abstract ***In this paper we present an innovative prototype Open Source Teaching/Learning Collaboratory created at UC Merced that will provide the foundation for offering the vast majority of our Computer Science and Engineering (CSE) courses, as well as courses from across our engineering disciplines, and, increasingly, computer courses and computer intensive courses throughout our university. This prototype facility is in operation at the present time, with one and possibly two additional Collaboratories to be installed in time for our Fall 2007 session. In addition, this facility will provide the model and framework for other local and distance teaching activities planned for supporting curriculum development and computing training. We show that by relying on free open source software and commodity hardware we can build an education computing environment that minimizes administration tasks, provides effortless remote interaction, simplifies and enhances the computer science curricula that can be delivered to the students and minimizes both the aquisition and operation costs.***

Keywords: collaborative learning, computers in classroom, case studies.

1 Introduction

The University of California at Merced opened in September 2005 as the first research university to be built in the United States in the 21st century. The newest of the ten campuses of the UC system provides unique opportunities to create novel and innovative pedagogical programs from the ground up. This paper describes one such effort: the Open Source Teaching/Learning Collaboratory that has been designed and built by the faculty and staff of the School of Engineering. The Collaboratory represents an effort to provide an effective, relevant and flexible environment for educating the next generation of computer literate and technologically confident college graduates.

We believe that the following notable features make the Collaboratory a model for educational computing environments of the future:

- Complete reliance on Free and Open Source Software (FOSS)
- Minimal reliance on administrative intervention
- Commodity hardware
- Effortless remote access/interaction
- Effective student-to-student and student-to-instructor interaction
- Cost effective in terms of both initial hardware/software acquisition as well as operational costs

The remainder of this paper is organized as follows. Section 2 provides context by describing similar systems. Section 3 describes the technical details of a prototype that is currently in operation at UC Merced. Section 4 describes some of the educational advantages provided by the Collaboratory. And, finally, Section 5 concludes.

2 Related Work

The central innovation of the Collaboratory involves the broad flexibility and accessibility afforded to students (and instructors) at virtually all levels of the laboratory environment. Many component technologies are needed to provide this open environment, and none of these components are particularly novel in isolation [1]. While a primary feature of this laboratory is the comprehensive use of open source software, this is certainly not the first teaching environment to make use of such

Figure 1: UCM Collaboratory. The pictures show students working on the Collaboratory doing their homework. The tables can hide the 24” monitors together with the keyboards to leave a clean and flat working surface if required.

tools. The utility of using open source software to teach computer science topics is well documented, with examples stemming from curricula in software design [2], operating systems [3], networking [4], security [5], and robotics [6]. Also, for many years, common courses in compiler design have made use of open source tools. The Collaboratory not only makes selective open source software tools available for course work, as these previous efforts have done, but provides students with an open architecture from top to bottom. The primary benefit of this configuration is the power that it provides to students, teaching them not only how to use open source tools, but how to modify them to fit their needs. For example, unlike previously investigated isolated virtual environments for granting students with kernel-level access within the context of operating systems coursework [3], the Collaboratory provides students with access to the actual kernel running on their laboratory machine, with automated support and security provided externally. Thus, students learn the nuances of manipulating actual kernels in real computing environments, rather than in pedogogically protected ones. Importantly, this ability is not limited to students enrolled in operating systems courses, so students studying robotics, for example, are able to safely make kernel modifications as part of their course projects. This kind of broad accessibility is a cornerstone of the Collaboratory concept. While other teaching laboratories have provided support for remote desktop viewing in support of such actitivies as distance learning [3], the Collaboratory encourages interested students to actually modify the systems supporting such networked communication, allowing them to help shape the very structure of educational communication and collaboration.

3 The Collaboratory

3.1 Overview

Our vision for the Open Source Collaboratory has been guided by the overarching goal of providing students a learning environment with minimal barriers. Our intent is for students to be able to manipulate their computing environments as needed without relying on administrative intervention and without running into administrative constraints. Students should also have tools available that will enable them to interact effortlessly with other people in the lab and beyond. Computing resources in the lab should always be viewed as tools and not hindrances. Part of achieving these goals will be complete reliance on Free and Open Source software (FOSS). Not only will this provide complete freedom in developing and maintaining our software systems, but it will enable transparency so that students (and others) can study and extend the capabilities we have developed [1].

The Collaboratory presently supports sixty seats for use by students during class instruction, and also for use outside class meetings, such as for completion of homework, research project involvement, and individual and group computing projects. The lab is also available for students in our Engineering Service Learning courses, and also for non-Engineering courses. The driving considerations in designing this lab were: (1) significantly enhanced usability, (2) strengthened security, and (3) efficient and effective systems maintenance. Prominent in this facility are net-booting, low-power computing, and VNC [7].

Each seat in the labs is set for net-booting from a single boot-server. All machines are essentially identi-


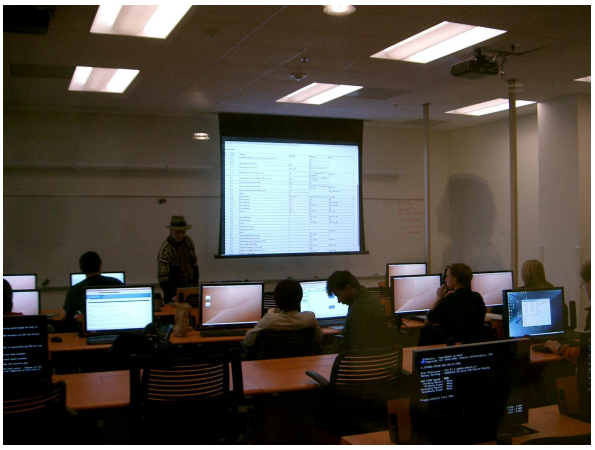

Figure 2: UCM Collaboratory. The pictures show students interacting with faculty before and during a lecture.

cal, booting with a built-in PXE loader in roughly 73 seconds. An individual workstation acquires read-only NFS-mounted working disk space during boot-up. As part of the boot process UnionFS uses tmpfs to create a read-write root filesystem. Once a user logs in–another 23 seconds–she sees her home filesystem in /home/username, a link to the mount point.

The design of these labs and workstations makes it safe and even natural to give all students logins, including even guest@localhost, privileges to sudo, access audio, and otherwise allow use of all the thin client capabilities. There's no harm in allowing–encouraging–root logins, because there's no local storage and each seat can be rebooted back to its default state, and quickly. Security limits the potential damage done by a hooligan with root at an individual seat.

Figures 1 and 2 show the use of the Collaboratory by UCM students and faculty.

3.2 Features and Functionality

The result is that students can walk in to the lab at any time and access:

- their usual $HOME; or
- they can treat the lab seat as a stand-alone box for installation of specialized engineering software; or
- it can be used as a highly-capable display for viewing remote scientific computations; or
- it can become a collaboration hub for group projects by local or remote students; or
- it can be a stand-alone access point for connecting to the Web.

Campus-wide logins are not resolved directly by LDAP [8], but sent by way of ssh to a dedicated machine on the LAN, which itself knows how to authenticate requests against LDAP and automatically create corresponding accounts. This has proven to simplify configuration of the lab seats and maintenance of the dedicated ssh-LDAP host.

Significantly, SSL-for-VNC leverages these other elements. X11-based displays, among others, can be viewed remotely and securely through a Web browser. This technique further reduces the learning curve or "activation energy" for a student or professor who wants to share a computation, even for a one-time-only or limited-use URL.

In any of these roles, all sessions and displays are accessible remotely with screen and VNC, so technical support can be provided swiftly and with great insight. In the rare event that a host breaks, or the more likely one that there's a need to scale up, it takes seconds to pull out another Mini-Box, attach power, keyboard, video, mouse, and network connections, and boot it, fully ready for full use [1]

The uniformity and utility of this variation on thin-client "utility computing" has liberated attention for less common benefits. With each seat serving screen and VNC, technical support is both simplified and physically decoupled. It takes fewer assistants to maintain operations, and many times they need not even be in the lab. As it becomes comfortable to move logins, displays, processes, and desktops around, instructors and teaching assistants will find it advantageous to check in on student work "live." Two workstations have been dedicated to big-screen projection and other multimedia

[1] The benefits are so evident, and the costs so low, that other facilities on campus have begun to ask for their own inventories of these devices.

playback, with two more scheduled for the future. It's easy to quickly set up shared displays on nearby workstations, or send them to the projector in the lab or a classroom. A student can develop a presentation in the lab, then initiate it interactively for his whole class. And because VNC clients are themselves ubiquitous, it is a simple matter for a scholar upstairs or across the country to make a demonstration available for those working in the labs.

Most of these features take only a few lines of coding and configuration. This project, for instance, created a standard way to securely forward a display. It includes only a few elements unimpressive in isolation, but, as near as we can tell, documented nowhere else in this especially useful combination.

Most exciting of all are two qualitative distinctions that are more academic than technical. First, usability helps bring the focus of students back to content rather than computing technique. A simple, uniform, and flexible computing environment eliminates the former premium on memorization of esoteric command-line or "wizard" sequences. Students can concentrate more effectively on their work, without such distractions as malware-infected hosts, broken parts, authentication or synchronization misconfigurations, and all the ills now evoked by the jargon word "silo:" value locked up in a particular machine or process, unable to communicate across boundaries. Emphasis on simple, reliable approaches makes for a significantly different end-user experience. Beyond the confines of the lab, "utility computing 2.0" bears on ambitions we have for scientific computing more broadly [1]. Scientific publications frequently refer to digital calculations. It's astonishingly rare, though, for scientists to make their source code available so that others might reproduce and more deeply analyze their results. Even the few scientists who try to do so–who think they're disclosing their source–often have a poor grasp of the computing commonplaces involved in portability and transparency, with the result that they inadvertently hide crucial information about their environments and configurations.

Appropriate "commoditization" of computing environments and thoughtful reliance on open-source products both act to make it easier for scientists and engineers to calculate the results they're after in ways that others can reproduce and monitor. At UCM and elsewhere, we're starting to see such payoffs: scientists who think of it as natural to share and publish not just their theories, commentaries, or data, but also the computing processes and displays that yield those results.

Item	***Minibox***	***Desktop***
CPU	$275	$1,100
24" monitor	$747	$747
Mouse&keyboard	$36	$36
1 GB memory	$80	$180
Total cost	$1,138	$2,063
Power drain ($)	$275	$1,100

Table 1: Comparison of the cost and efficiency of Collaboratory "seats" to those typically found in conventional teaching laboratories (from Laird and Laird [1]).

3.3 Costs and Efficiencies

In contrast to a conventional computing lab, the lab is using ITX Mini-Box M200s driving 24" Dell LCD displays set to 1280x1024, which is not only extremely cost effective, but very energy efficient as reflected in Table 1.

Total cost-per-seat is a modest $1,138 almost half that of a conventional lab seat ($2,063). Also, the savings of about 180 average watts represents, at prevailing California electricity prices, around $10 per month, or well over $100 per year. That advantage more than triples when one considers that the power savings also represents a reduced load on the air-conditioning facility for the labs. In reality, savings on the CPU make the big 24" displays feasible. Our thin-but-wide clients are very popular with student users.

4 Educational Experience

We think the Collaboratory is an invaluable tool for introducing new computing concepts at all levels of the computer science and engineering instruction, from the first introductory level computer science courses to the more advance upper division and graduate level courses. The following sections provide some simple examples of our vision and goals from the education point of view.

4.1 Introductory Computing

Two aspects of our introductory computer programming curriculum are particularly well suited to the tools provided by the Collaboratory environment. First, unlike many programming courses that involve lengthy lectures and little oversight during hands-on programming activities, the UCM curriculum includes weekly multi-hour sessions of instructor-supervised laboratory experience. Substantial time is devoted to the practice of source code generation in a context that allows for interactive guidance by the teaching team. The structure of

the Collaboratory allows student desktops to be dynamically selected for presentation before the class, supporting the rapid sharing of pedagogical insights garnered through interaction with individual students. For example, a conversation with a given student might uncover a misunderstanding that might be shared by other students, and the instructor has the option of displaying the desktop environment in which the misunderstanding arose in order to clarify issues for the entire class. Remote access to student desktops also facilitates one-on-one student-teacher interactions when the instructor is not physically present in the Collaboratory, making it easier for teachers to assist students even from a distance. A second aspect of the Collaboratory that is well suited to the UCM computer programming curriculum involves the degree to which experience with practical software development tools is incorporated into classwork. Our intent is to provide students with training in modern programming practice, including the use of integrated development environments (e.g., Eclipse) and source code version control tools (e.g., CVS). The curriculum is designed to foster the development of skills that might eventually lead to the completion of substantial capstone projects by computer science majors, such as contributing to a real open source software development effort. The Collaboratory is well suited to allow students to experiment with distributed software development, providing them with the tools needed to establish CVS repositories, for example, and manage access to such source libraries.

4.2 Artificial Intelligence

The UCM course on artificial intelligence (AI) is currently being redesigned to leverage the strengths of the Collaboratory. In addition to making use of the ability to share individual student desktops with the class, the Collaboratory offers a particularly novel opportunity for teaching techniques from the distributed AI literature. For example, so-called "blackboard architectures" or "pandemonium architectures" for tasks such as speech recognition involve many interacting parallel processes, each of which performs some small part of a larger task. By having each student implement only one of these processes on their own machine, and by having the processes interact over the Collaboratory's network, students can see how the seemingly chaotic competition and cooperation of the various processes gives rise to a coherent result. During such demonstrations, teachers may use the Collaboratory's desktop projection tools to dynamically display the activity of the particular processes to the full class as the distributed system performs its task. The open network architecture also provides support for other explorations into multi-agent AI, such as competitions between game-playing programs designed by individual students without a need to transfer source code between students.

4.3 Computer Graphics

The Collaboratory will support the development of new courses in computer graphics oriented to the design and implementation of video games. The video game industry has tripled its sales in the last decade to become a $7 billion industry. This growth, together with student interest, has caused several CS programs across the country offer courses and even majors related to computer games. The flexibility of the Collaboratory to provide remote access to sessions and displays will in particular support students to work on project implementations related to multi-player games, study the scalability of the implemented systems, and understand the issues involved in massively multi-player online games. A Collaboratory ready to explore such computer game topics will attract more students to the school's computer science programs, and in particular bring very motivated students.

4.4 Computer Networks, Operating Systems and Distributed Systems

The Collaboratory is an excellent tool for teaching computer networking and distributed system concepts. The distributed nature of the Collaboratory, together with the ubiquitous display capability allow the study of highly complex topics in distributed algorithms using real networking interfaces, sending and receiving real network traffic instead of just using a single machine simulator with artificially generated traffic. Students have the chance of understanding and measuring the "self-similar" nature of local area traffic, and understand concepts like congestion control in the face of competing traffic. One of the most important features of the Collaboratory is the capability of booting any operating system without imposing any security risk or damage to a local image of the operating system. This feature is critically important for some courses, in particular for the introduction of operating system concepts. Students can boot any OS, gain "root" access and even start modifying the kernel without imposing any security risks; if a catastrophic error occurs a simple reboot solves the problem and the machine becomes usable again with a fresh image, without any hassle or tedious system administration tasks. Perhaps more importantly, the Collaboratory itself presents an excellent case study for the design of distributed systems and applications, using an elegant self-referential design for the study of distributed systems concepts.

4.5 Image Processing and Computer Vision

The visual nature of courses in image processing and computer vision makes them very popular with computer science undergraduate students. Students are captivated by being able to literally see the results of applying the algorithms-this provides great motivation. The powerful, dynamic visualization capability of UC Merced's Collaboratory makes it an ideal teaching lab for courses in visual computing. During a lab session, the instructor can selectively display a particular student's visual results to the rest of the class using the projectors and the VNC sublayer. This allows for a richer learning experience as, for example, the whole class can observe the effects of different parameter settings for an algorithm without having to physically move around the classroom, crowding around individual displays.

4.6 Mobile Robotics

If we consider a class introducing students to mobile robotics topics, the availability of virtual shared place where different students can interact is of utmost importance. Indeed, the use of simulators is greatly beneficial when teaching such classes. The Collaboratory can serve as shared infrastructure where different student teams can instantiate their simulated robots and see how they perform when interacting with autonomous agents deployed by others. While this objective is in fact obtainable with a straightforward client-server simulator accessible via the Internet, the Collaboratory adds the possibility for the different teams to interact in real-time and to exchange immediate feedback to discuss the reasons behind observed behaviors. This aspect is of overwhelming importance when studying complex systems like multi-robot systems, where the noticed behavior is the result of intricate interactions that are not necessarily evident to the learner.

5 Conclusions and Future Work

While our Collaboratory has already proven extremely valuable in supporting our rapidly emerging open source culture and our innovative new courses, there's plenty of work left to do. The facilities for session recording remain incomplete. We're exploring ways to accelerate display of remote sites, and building tools to simplify naive use of distributed resources. Security is, as always, a never-ending challenge. Quite a few small tuning opportunities remain: further reductions of boot-time; use of panning or scaling on the projectors to match the potential 1900x1200 resolution afforded by the workstation monitors; acceleration or replacement of VNC with NX or other technologies; integration of VoIP functions; innovations in collaborative processes; and automation of outbound VPN to peer sites are a few of the projects we might tackle.

References

[1] Kyler Laird and Cameron Laird. Remote computing with a linux application server farm. *IBM developWorks*, Feburary 6 2007. http://www-128.ibm.com/developerworks/linux/librry/l-server-farm.html.

[2] David Carrington and Soon-Kyeong Kim. Teaching software design with open source software. In *33rd ASEE/IEEE Annual Conference of Fronteiers in Eduaction (FIE 2003)*, volume 3, pages 9–14, 2003.

[3] Jason Nieh and Chris Vaill. Experiences teaching operating systems using virtual platforms and linux. *SIGOPS Operating Systems Review*, 40(2):100–104, 2006.

[4] Daniel Nelson and Yau Man Ng. Teaching computer networking using open source software. In *ITiCSE '00: Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE conference on innovation and technology in computer science education*, pages 13–16, New York, NY, USA, 2000. ACM Press.

[5] Prabhaker Mateti. A laboratory-based course on internet security. In *SIGCSE '03: Proceedings of the 34th SIGCSE technical symposium on computer science education*, pages 252–256, New York, NY, USA, 2003. ACM Press.

[6] Keith J. O'Hara and Jennifer S. Kay. Investigating open source software and educational robotics. *J. Comput. Small Coll.*, 18(3):8–16, 2003.

[7] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[8] Vassiliki Koutsonikola and Athena Vakali. LDAP: Framework, practices, and trends. *IEEE Internet Computing*, 08(5):66–72, 2004.