The Right Tools for the Right Jobs: Developing a Student Management System

J.P. Brannan
Cornell University
Computing & Communication Center
Ithaca, New York 14853
(607)-255-2523
jpb1@cornell.edu

James Landis
Cornell University
Computing & Communication Center
Ithaca, New York 14853
(607)-255-8990
jcl24@cornell.edu

ABSTRACT
Many campus IT organizations rely heavily on student employees and part-time employees in HelpDesks, Labs and other work groups. Though each type of group may be managed differently, managers face some common tasks:

- Hiring a traditionally high-turnover work force
- Tracking employee training and expertise
- Scheduling part-time employees
- Tracking time worked by part-time employees
- Keeping records of employee performance
- Managing how part-time employees can swap shifts with others
- Doing it all cheaply and effectively without taking up a lot of time

This presentation will describe how Cornell has met these challenges by implementing a student management system that can be used by multiple groups on campus with minimal cost. Relying on open source solutions including Linux, Apache, PHP and MySQL, Cornell has successfully deployed a modular student management system that allows departments to pick and choose which modules they use, including:

- On-line job applications and hiring processes
- Employee information tracking
- Setting a “standard” employee schedule
- A tool allowing employees to swap shifts
- On-line management of schedule parameters (hours of operation, number of employees, etc.)
- Employee performance records

The presentation will also describe the issues involved in designing and implementing a student management system and how Cornell addressed them, deploying the system with a web interface within 10 weeks of the project’s inception. It will illustrate how various “home-grown” administrative tools were combined and shared across departments, saving staff time, standardizing and streamlining administrative processes.

Keywords
Student management, scheduling, web interface, php, SQL, MySQL, Linux.

1. INTRODUCTION
Many campus Information Technology (IT) organizations include student or part-time employees. Many student employees are used in the service sector, as computer lab operators, help desk staff, field support consultants, or trainers. Some students may also be employed as programmers, web designers or analysts. Regardless of the specific tasks assigned to each student employee, IT managers must deal with the same core challenges when supervising student employees. Students are usually available to work according to an academic schedule, which means managers must not only work around class schedules, they must allow for portions of their work force to be unavailable during academic breaks, to sometimes take entire semesters off and eventually to graduate and leave altogether. In addition to scheduling concerns, a seasonal, high-turnover student work force creates a need for the ongoing recruitment and training of employees. It also makes tracking the group’s technical expertise harder and generally increases a supervisor’s record-keeping burden, especially when tracking employee time.

At Cornell University, managers in Cornell Information Technologies (CIT) recognized that several groups within CIT had the same needs: the HelpDesk, ResNet and the public labs, for example. Other groups, including the Publications and Information Group (PIG), Unix and Web Services (UWS) and the Academic Technology Center (ATC) had at least a subset of those core needs. The development of a common, standardized application for handling administrative tasks like tracking job applications, tracking employee knowledge and training levels,
tracking employee performance and scheduling has minimized the duplication of administrative efforts and helped make CIT’s overall management of student employees more effective and efficient.

2. DEVELOPMENT PROCESS

The process CIT used to develop its Student Management System (SMS) evolved based on a deadline for product rollout, which in turn was based on student programmer availability. The project began in late November 1999, and had the goal of producing a usable system by the start of Cornell’s spring semester in January 2000.

2.1 The Idea

The project was inspired by an analysis of existing departmental administrative systems and processes. CIT’s HelpDesk and ResNet groups were using several different technologies to perform related tasks. Shift scheduling and swapping were handled by an SQL database and PHP3 scripts being served by an Apache web server on a Linux system. On-line jobs applications were being stored as static HTML pages generated by a Perl CGI script being served through Apache on a Solaris system. Employee information and time were being tracked by FileMaker Pro databases being served over the World Wide Web by WebStar and Tango for FileMaker on a MacOS system. A review of the cost of maintaining the various systems, both in terms of time and money resulted in a recommendation to consolidate the systems into a single server and technology base. Given the flexibility of SQL and PHP3, the project team decided that all of the tasks could be handled by a PHP/MySQL system. The stability of a Unix system was desirable for the server, and cost pressures led to the use of Linux as the operating system of choice for the system.

2.2 Need Assessment

Once a technology base was established, the project's lead developer conducted a series of interviews with potential users of the system, mainly HelpDesk management. The interviews allowed users to describe what functionality they wanted in the new system. It also allowed the developers to identify shortcomings in the pre-existing systems. During this phase it quickly became apparent that the consolidation of the various legacy systems would allow the new system to have features that were greater than the sum of the features of the previous systems. In some cases this was because one system would have access to all of the data, which allowed for more complete and complex reporting. In other cases it was simply the case that previously difficult tasks would be relatively simple to perform with the new technology base. This phase resulted in a mock-up of the new system on a series of white boards. It was decided not to re-use any code from the older systems in the creation of the new system.

2.3 Building the Prototype

Once the system's specifications were mapped out on white boards, it was time to actually build a working model. The goal of this phase was to make a prototype that could be shown to a major user group, in this case the HelpDesk and ResNet staff. The prototype could then be tested by users and modified as needed before final deployment. In order to meet time constraints development of the prototype was broken into three tracks, which were each developed in parallel.

2.3.1 Data Design

The lead developer designed a MySQL database to fit the specifications produced in the Need Assessment phase. He then began coding the php3 scripts that would manipulate the data and produce the graphical user interface. This was the longest single piece of developer time, approximately three to four weeks. One developer handled all of this work. It might have taken less time if split up among two or more developers, however the single-developer approach allowed for quick changes in direction as plans were changed.

2.3.2 System Administration

While the lead developer created the database and scripts on his workstation, the project's system administrator built the system on which the finished product would run. The operating requirements of the system did not require intensive system resources, and the strategy adopted by the system administrator was to acquire two reasonably cheap (approximately $1300 each) desktop-level systems, with one system being server and the other serving as a redundant backup system in the event of hardware failure. The computers chosen were Dell Optiplex Pentium III-500 systems. This decision was based primarily on cost and existing purchase agreements between Cornell and Dell Computer. Any Pentium-class Intel-based system would have sufficed. Once the hardware was in place the system administrator installed Mandrake Linux, MySQL, Apache, SSHD2 and PHP3. Installation took perhaps a week, with configuration of system security and access taking another week.

2.3.3 Authentication and Authorization

User authentication had been a major challenge for developers of previous systems, especially when developing for MacOS and/or Microsoft Windows servers. The selection of a Unix-based operating system allowed the developers to leverage Cornell's existing Kerberos 4 user authentication infrastructure. Other developers at Cornell had already produced software that allowed Unix-based systems to authenticate users via Cornell's Kerberos server, and had even gone so far as to include that ability in an Apache module. The major challenge involved compiling those tools for Linux and rebuilding the Apache module to work as a dynamic server object. The authors of the applications needed to be called in at one point, but overall this part of the development took less time than the rest of the system installation and configuration. Once authentication needs were met, so that system could be sure of a user's credentials, a simple authorization scheme was put in place using an existing campus-wide authorization server integrated with the Kerberos system. There are six hierarchical levels of access within the Student Management System: guest, user, user administration, supervisor, manager, and system administrator. Although much of the authentication and authorization depends on Cornell's computing infrastructure, the authentication/authorization component could be adapted to use other technologies, as long as the same access-level hierarchy was used.

2.4 Pilot Deployment and User Training

Once the prototype had been assembled, tested by users and revised based on user feedback the Student Management System
was deployed in two of the intended user groups: CIT's HelpDesk and ResNet support groups. Because the system was designed based on user requirements and the prototype interface had been modified based on user feedback the system was fairly intuitive and the need for user training was minimal. This is not to say that there was no need for training. Users of more advanced features, like supervisors and managers required more training than average employees, who had a fair amount of training for the system. The system was deployed in January 2000, and was used throughout Cornell's spring semester. Some modifications had to be made along the way. The hiring and reporting modules, being the newest additions to the administrative suite, needed more adjustment than other modules. This pilot phase continued through May 2000.

2.5 Adjustments for Full Deployment

Once the system had proven successful in limited deployment, the project team met with representatives from each user group and evaluated the readiness of the Student Management System for deployment beyond the pilot deployment areas. The SMS hiring module was deployed in CIT's training, PIG, ATC and UWS groups, and the hiring module was integrated with CIT's public World Wide Web server. Although the hiring module had been accessible via the web server during the pilot phase, it was largely due to manual intervention on the part of the webmaster. In the full deployment phase, web pages were generated dynamically, letting the public know which positions were available at any given time without manual webmaster intervention. SMS was also introduced to CIT's public labs group. The labs group would be using more of SMS than the groups mentioned previously, and as such they required additions to the reporting functions. Further efforts to synchronize the group's business processes with those of other groups were also needed. To meet those needs and to match Cornell's academic calendar, deployment in the labs group was delayed until August 2000 and a programmer from the labs group joined the project team.

2.6 Development of Additional Modules

Since the initial deployment, two new modules have been suggested and are in development. One module tracks the skills of student employees, so that CIT managers, even those in other groups, can find experts in given areas more easily. The skill module would also allow managers to identify gaps in employee training and develop new training programs as needed. The second module tracks student employee payroll. A payroll module was not included in the original design because Cornell already had a campus-wide student employee payroll system. However, the three groups using SMS could benefit from additional historical tracking of student employee payroll that is beyond the capacity and scope of the campus-wide system. Development of these modules is ongoing as of July 2000.

3. DEVELOPMENT ISSUES

The issues involved in the development of the Student Management System are similar to those found in the internal development of any administrative application. This effort differed from other administrative application efforts in that it was an attempt by a few service-sector computing groups to meet their own administrative computing needs without formally involving CIT's programming groups. The project was initially funded and resourced by the HelpDesk and ResNet groups. The development team was learning much of the technology they were using as they used it. Having the development team in-house for the pilot deployment helped in two important ways. First, users were more willing to give applications a chance when they knew the developers who had created the system from scratch, and were more willing to offer constructive feedback. Second, because the developers were in-house to witness all the feedback firsthand, the development was able to be more responsive to user demands than any outside developer could have been.

3.1 Time

The major priority for the project team was to deliver a working product on time. The initial evaluation and interview phases used more staff time than might have been necessary, because many staff interviews were conducted. The larger amount of time spent on the initial assessment was intended to reduce the amount of time that would have to be spent later revising the system. The parallel development process required more developer time over a shorter period than serial development would have. The extra staff time required during the November to January development period was deemed acceptable given the need to push the process to completion.

3.2 Money

Beyond the cost of staff time in developing and testing the system, the startup cost was low. Each server cost $1300, and RAM was later upgraded in each system for approximately $150 each. Developer workstations were assembled from surplus equipment. The Linux operating system was free, as was the Apache web server. The PHP scripting software, MySQL database software and other system utilities were all free. The only software cost was for Secure Shell (SSH2) clients that allowed developers to login to the server remotely and securely. The initial cost for SSH2 clients was $100, though later more licenses were purchased. The cost of Tivoli ADSM tape backup for both servers worked out to be less than $30 per month. The major cost associated with the system is the internal cost of keeping the primary server in CIT’s machine room where it is monitored by operations staff 24 hours per day, seven days per week. The estimated yearly cost of keeping a server in the machine room is $1550.

3.3 Maintenance and Ongoing Development

Limited programmer time made the development of administrative tools critical. System information and parameters are stored in the system's database. Administrative tools allow authorized users to alter parameters without the need for programmers to change hard-code variables. This reduction in the need for "maintenance programming" allows programmers to concentrate efforts on development. Some time is spent on upgrading various software components (operating system, web server, php, etc.) as new versions are released. However, before any upgrade is installed on the primary server it is first tested on the redundant backup server. The modular design of SMS makes adding functions as easy as adding new modules. New modules can use existing shared modules, such as the authentication/authorization module, which is shared by all other modules. Existing modules can be replaced or upgraded as needed without altering the rest of the modules. The design team itself is modular. As other groups start using SMS heavily, part of the process will undoubtedly involve the inclusion of representatives
from the new groups on the design team, joining the established core group.

4. ACKNOWLEDGMENTS
Our thanks to James Byers, Amanda McAuley, Celisa Manly, Chris Manly, Lisa Mix, Jim Lombardi and Annie Stunden.