Labtool - A Managing Software for Computer Courses

R. Heinzl, G. Mach, P. Schwaha, and S. Selberherr

Institute for Microelectronics, TU Wien
Gußhausstraße 27-29/E360, A-1040 Vienna, Austria
E-mail: {heinzl|mach|schwaha|selberherr}@iue.tuwien.ac.at

KEYWORDS

computer courses, managing software, monitoring and grading system, web front end, `LDAP`, `kerberos`, `PHP`

ABSTRACT

A comprehensive tool to manage computer courses was developed relying on a library centric application design. Backward compatibility to existing systems, and special emphasis on usability were also part of the design process. The experience gathered from implementing monitoring and controlling modules for the heterogeneous hardware and software system was applied to make the system error tolerant. To this end we specified, developed, tested, and evaluated several modules and a graphical user interface and finally integrated them into a programming computer course at our university.

INTRODUCTION AND MOTIVATION

The basic question we faced at the beginning was, how to automatically organize, e.g. a programming language course for 200 to 300 students in groups to 30? More importantly, how can a heterogeneous system with various services and tasks be implemented in a stable fashion that does not interfere with a continuing evaluation of students? We therefore summarize the most important issues in developing software for computer courses from the view of a programming language lecture for the first and second semester at a university:

- availability of the system needs to be at 100 percent during the courses
- automation of various tasks such as tracking the progress of students, report generation, and system service overview
- limited resources such as instructors, and computers
- appointment constraints like other courses (on both sides - students and teachers)
- necessary data exchange with other departments of the university or other programs to administrate the course

Since we did not find any free graphical tools to avoid excessive paperwork and security holes in a modular, redundant, backward compatible fashion with the administrative databases, we decided to develop and implement our Labtool. Not only the finite resources but also social aspects had influence on the specific design. We present our implementation, focusing on the encountered problems and their solutions.

A GENERALIZED APPROACH

One reasonable solution to the problem of limited computer resources and a great number of students is to split the students into groups of fixed size. Each of the groups then has several fixed dates assigned and students are free to choose one of the groups. The advantage of this approach is that the group size is fixed and that the memberships and appointments are easily coordinated with other courses. The disadvantage is that individuals cannot change the dates associated with the group they belong to. If a student cannot attend at a set time, either individuals need to be allowed to change between groups on single occasions or changes of group membership as a whole have to be considered. This, however, eliminates much of the management simplicity which is among the original benefits of this method. Not taking this into account will result in students not being able to finish the course due to time-related reasons.

Another approach is to let the students choose individual appointments from a predetermined list of dates for each unit of exercises. More dates need to be listed than in the previous approach to accommodate all students, but on the other hand students have a large degree of freedom and flexibility in choosing their appointments.

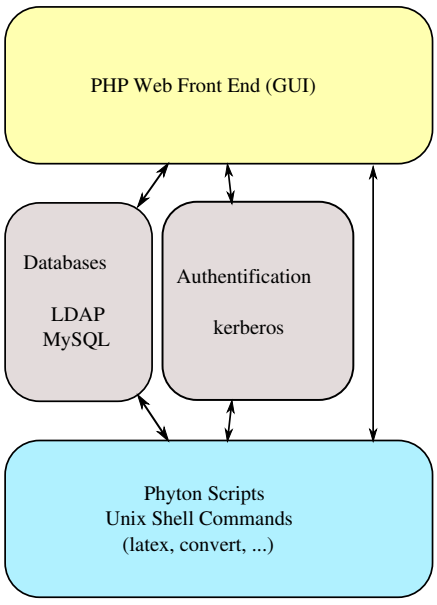

Figure 1: Based on an orthogonal software design the Labtool can be split into several modules with consistent interfaces.

An important fact related to social issues is the incorporation of the users. Whenever developing and implementing new management software, it turned out that acceptance among users is highest when

- they are involved in the development process and
- the software is fully backward compatible to any solution that may have been deployed before.

The integration of users in the whole development process not only increases acceptance, but also reduces the probability of erroneous use of the whole system.
The remaining issues listed in the requirements can be addressed by technical solutions which are presented in the subsequent sections.

THE DEVELOPED SYSTEM

The complete system consists of a laboratory cluster with several clients running basic services connected to two servers and the management software running on these servers. A brief overview of the structure of the management software is given in Figure 1 whereas an overview of the technical infrastructure is given in Figure 8.
The management software is composed of the following modules:

- a graphical user interface written in the dynamic web programming language `PHP` [7],
- user databases implemented in the Lightweight Directory Access Protocol `LDAP` [3],
- `python` scripts to produce lists in `LaTeX`,
- an authentication module implemented by the computer network authentication protocol `kerberos` [6] and
- a monitoring module to observe the status of system.

A very important part for the acceptance of such a system is the ease with which the most often used tasks can be executed. The whole system has not only to be easy to understand and use, but also has to be presented appealingly. The user has to be comfortable when working with the management software. We therefore put special emphasis to make all often used tasks available with as few clicks and inputs as possible.
Figure 2 shows the 'Enable' mask which allows to enable user accounts for students already registered for the current date as well as to join students who have not done so (register them for the current date and enable their user accounts). The input mask at the top provides an easy to use facility to tag students with specific unique identifications, names or registration numbers or parts thereof. Most of these functions can be controlled by various click functions, e.g., it is possible to enable the accounts of students with a singe click in the 'Enable' mask.
Also, we paid special attention on the concept of orthogonal application design: Each module can be used by itself or in combination with the graphical user interface (GUI), which furthermore makes it easy to modify, supplement or substitute single components.
In the following we present the main features of these modules. Authentication via `kerberos` is used to login to the Labtool. This has the advantage that we did not have to implement and maintain our own user management, but were able to use the data already provided by the central university system. Additionally many of the functions require the Labtool user to be known to the system, because only certain users have sufficient privileges to perform some tasks. We therefore implemented three groups of Labtool users:

- normal users, the instructors to hold the exercises,
- admin users, the managers of the course to additionally modify master data of the students, dates, and appointments, and
- system administrators able to monitor and restart services and the firewall.

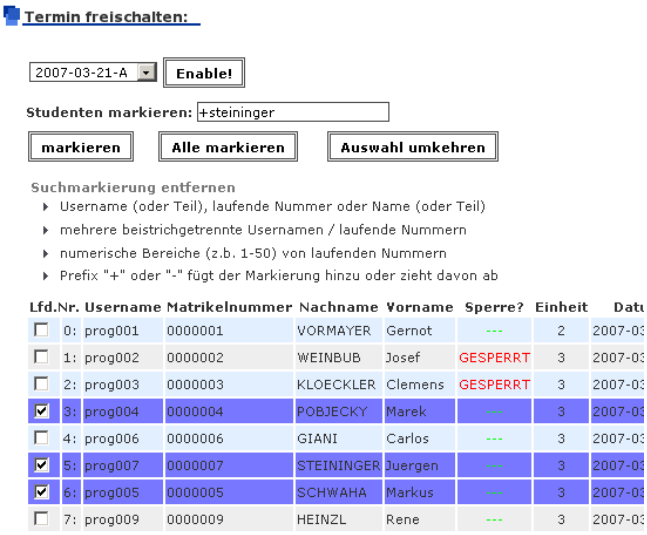

Figure 2: Registered students are automatically preselected. Additional students can be added to and removed from the selection using the input mask.

Another design decision was how to store user data in a database. The facts that the central university system supports `LDAP` queries against its students database, but also because the tree structure enforced by `LDAP` maps the structure of the course best and is optimized for these accesses, were strong arguments for ranking `LDAP` higher then other common solutions such as `MySQL` [8]. The opportunity to check the registration data such as names or registration numbers against the central database of the university is important when verifying the identities at the begin of exercises and much more when transmitting grades to the department of reporting. The most difficult time is during the beginning of the semester, when the courses already start but the term of registration has not yet expired. It is not always possible to verify the names and registration numbers of some of the students reliably during this period. A manual management system is therefore also available.
A student has several attributes, depicted in Figure 3, such as names, sex, registration number, and certain flags concerning his status at the course and at the system. All students have to attend each unit of the course. Therefore the student subscribes to each unit at a certain date thereby generating an appointment. Each appointment is then evaluated. If a student shifts the date of an appointment or is absent for some reason, another appointment is made available in the tree. Otherwise, if an appointment has a valid

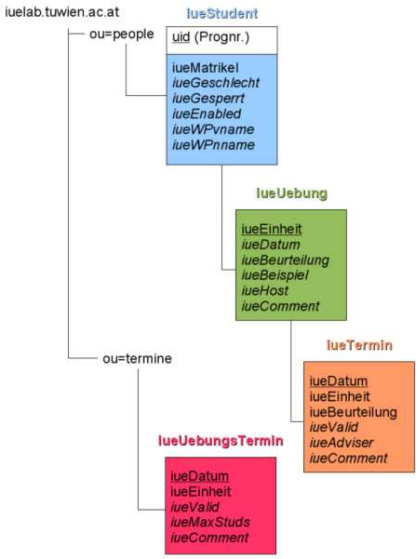

Figure 3: The LDAP-Tree shows objects and attributes we implemented. The attributes written in italics are optional.

grade (the student attended the exercise), the respective unit is evaluated. Figure 3 shows the student related information first, then the separate units, then the appointments, and finally the possible dates to subscribe for the units.
There are many reasons for keeping the possibility to print out reports of presence and success of a specific day or the overall list of the course. The most basic reason is to have a hard-copy backup in case the whole system crashes. Another reason would be the possibility to keep the course running, while the management software is temporary out of order. We already pointed out that acceptance also can be approved by keeping methods to easily downgrade to an older system due to the fact that people do not want to be tied to changes that cannot be undone.
For all these reasons and due to the concept of orthogonal application design we decided to implement `python` scripts and `LaTeX` templates to produce `postscript` files that can be stored and printed within and without the management software. Figure 4 shows an example for such a list for a single day.

LISTE PROGRAMMIEREN 2 (Ü3) 2007-03-21A

O	prog004	0000004	POBJECKY Marek
∀	prog005	0000005	SCHWAHA Markus
-	prog007	0000007	STEININGER Juergen
#	prog009	0000009	HEINZL Rene

Registered: 4
Missing: 1

Figure 4: The list of the day is produced with `python` scripts and `LaTeX`.

The graphical user interface front end is written in a `PHP` web application for many reasons:

- Every computer providing a web browser is able to control the Labtool,
- people are accustomed to web front ends,
- no necessity to implement a separate GUI, and
- `PHP` provides facilities to connect to all other modules (`LDAP`, Unix Shell Scripts, `kerberos`, `MySQL`, etc.).

For security and usability reasons the GUI is split in several sections (See Figure 5).

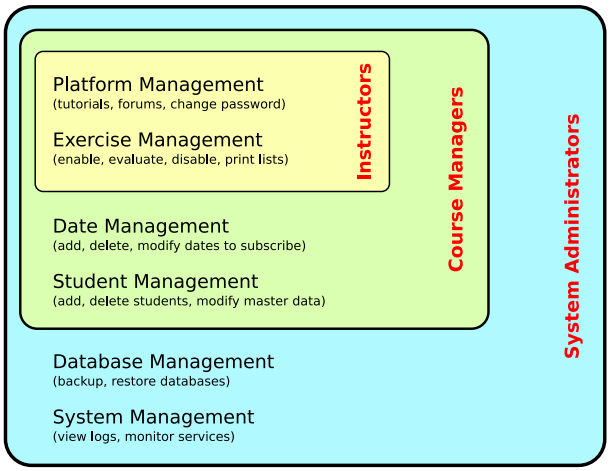

Figure 5: The GUI is divided in sections corresponding to the user groups.

The sections dealing with the platform specific tasks (tutorials, forum to post new ideas regarding the platform, password change functionality) and holding an exercise are available to all users. To further increase comfort and minimize needless output of paper we implemented a preview for all lists that may be printed. We use `ImageMagick` for the necessary conversions. Clicking a provided thumbnail opens a new window with the preview as shown in Figure 6.

The sections to control new dates to subscribe to and manage the students' master data are only available to managers of the course. They can add, delete, and modify the data of students, dates, and appointments. It is also possible for them to modify the grades of any student and any appointment in this section. The last part of the GUI is reserved for the system managers and provides logs, monitoring modules for the services and the firewall as well as the possibility to backup and restore the underlying databases.

One of the features to increase comfort for the user and the security of the system is to offer the possibility to easily undo changes or manage backups from different particular dates. This solves the problem of data loss in general and eases the handling of the whole system. A second reason for data loss is fragmentary transfer of input which can be avoided by redundancy.

Security is also threatened from outside of the system. Wherever computers have a high bandwith link to the world wide web, like at universities, the rates of attack can be considerable. Furthermore students attempt to get access to management systems in order to obtain information, change their grades, or try to get unfair assistance from other students or the Internet. For all of these reasons our Labtool is protected on the one side by a firewall and intrusion detection systems, and on the other side with additional database encryption in sensitive areas.

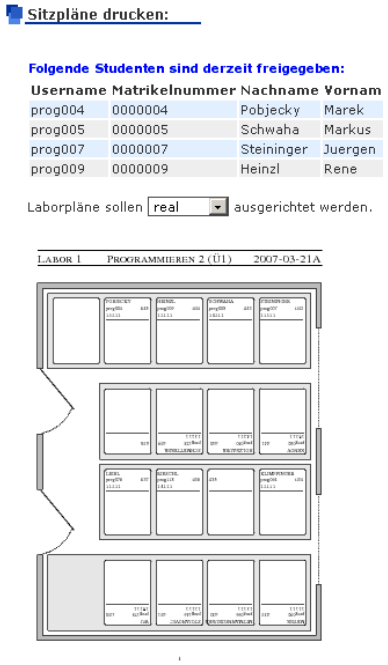

Figure 6: Clicking on the small lab maps opines a new window with the preview.

PROBLEMS AND SOLUTIONS

One of the the major problems we want to point out was the deficient realization of the `kerberos` functions in `PHP`. Because available documentation was badly lacking, our approach was to write our own `kerberos` library for `PHP` using the shell commands of `kerberos` and the `exec()` function of `PHP`.

Also, the system needs a great number of different services running, devices to be mounted, and systems to be synchronized. Therefore we not only implemented modules to monitor all of these services and states but also a kind of self healing mechanism. The monitoring modules ensure that system administrators can find sources of errors easily and quickly (see Figure 7).

The self healing modules try to avoid specific problems, such as:

- asynchronous system clocks of the clients or unreachable `NTP` service on the server,
- not mounted net devices on the clients or unreachable `NFS` service on the server,
- unreachable `kerberos`, `syslog` and `fcron` services regarding the system and
- unreachable `apache`, `LDAP`, `MySQL` and `cups` services regarding the Labtool

by periodically checking their status and attempting to restart the services, and synchronizing in case of errors.

The need to keep all system clocks of the cluster synchronized originates from the use of `kerberos`. It is due to the fact that `kerberos` uses timestamps as part of its authentication, the tolerance of clock skew lies within 300 seconds [6]. The situation gets even more difficult, when, for security reasons,

Figure 7: The current status of all important services can be monitored.

only the servers are able to connect to the Internet. The servers update their local clock from time servers outside the cluster and have to propagate this new time to all clients fast enough to stay within the interval of tolerance in order to keep authentication working. Whenever a client was offline, it first has to synchronize its time with the server before authenticate against `kerberos` is reliably possible.

To establish certain standards of security for the servers against threats from outside the lab cluster we implemented not only a firewall with `iptables` but also employ port scan detection (`psad, portsentry`) and intrusion detection tools (`samhain, aide`). Furthermore we installed the hardened kernel [5] sources and built a monolithic kernel to block as many routes for break-in attempts to the system as possible and alert the system administrators automatically via email.

On the side of the clients in the cluster we do not allow communication with hosts outside the specified range of IP addresses by not providing a default gateway. Due to the fact that `SHFS` is not a real distributed file system, its unavailability at boot time and because further utilities are needed in addition to the kernel modules for its deployment, we had to choose a solution from the following file systems:

- `AFS`, which uses only version 4 of `kerberos` (which makes it vulnerable to plain text attacks [2]) and the stateless UDP protocol (which makes it difficult to use with firewalls) and being only tagged as experimental in the kernel which we used.
- `CODA`, which has low speed of writing and deleting files, no mechanism to make files persistent other than closing the file (which makes it difficult to use for log-files and other files kept open for long periods of time), and requires that entire files have to be read [4] (which means that the cache must be at least as big as the biggest file will ever be).
- `NFS` version 4 [1], which uses the state-full TCP protocol instead of UDP, allows authentication of users via `kerberos` version 5, and is economic in its use of memory. In our experience it still suffers from a few stability issues and may even freeze the system, if any of the `NFS` related services fail.

To ensure minimal downtime of the system in case of a crash of a server we mirror all necessary services, such as:

- `NFS`, `kerberos`, `syslog` and `fcron` (system functionality)
- `apache` and `cups` (Labtool functionality)

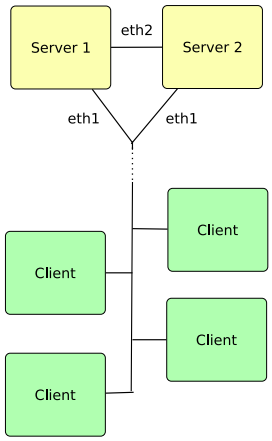

Figure 8: The cluster is connected via Gigabit Ethernet Connections.

and the following two databases:

- `LDAP` for the main databases
- `MySQL` for additional logging purposes

on a second server (see Figure 8). Furthermore all data from the students (i.e. home directories) and the Labtool (i.e. all lists) are backuped on a host outside the cluster every day.

OUTLOOK

We have several ideas how a system such as Labtool can be enhanced and extended and present them in the following. One possible extension is to implement methods that the success of students can be evaluated by the tool automatically. To this end it is necessary that the computer programs implemented by students have defined interfaces for input and output via the `main()` function. Having a catalog of stages with a defined interface in this manner, the Labtool could easily detect the stages already working. Currently the tasks of the instructors in our laboratories are not only to provide assistance to the students with their programming tasks, but also perform a final inspection of the students' programs and to evaluate their skills. Enhancing the software this way would decrease the amount of work of the instructors and increase objectivity of assessment at the same time.

Another improvement would be to handle the management of appointments completely within the Labtool. The advantages aside from saving work and time are first the lapse of transfer interfaces between different programs and therefore the omission of a source of error. Second the students would be able to do bilateral changes of appointments by themselves. A possible way to achieve this is to enable the students to send some kind of invitation to change an appointment such as the invitations sent by social software communities / social networking websites to either a specific student (for this the students must be able to see which student is registered for which date) or to all students registered for a specific date. The other student can then accept or decline this invitation. If a student accepts, the appointments are changed and all other students get a message that the invitation is recalled.

Implementing the feature this way would allow changes without being online at the same time or on a common interface, and no session keys would have to be exchanged.

CONCLUSION

We presented the development and implementation of a comprehensive management tool for computer courses. We thereby outlined the use of the orthogonal software design concepts to keep the modules of the system exchangeable. Special attention is paid to error tolerance, monitoring and controlling software as well as self healing mechanisms. The system had to be integrated into a running computer course with an emphasis on backward compatibility. Usability was always not only kept in mind, but was one of the paramount goals of the development. We also discussed problems that surfaced and possible ways to resolve them, so that an interested reader should be able to re-implement a similar system and to avoid most of the problems that typically arise.

REFERENCES

[1] *NFSv4*, 2005. `http://www.nfsv4.org/`.

[2] *Open AFS*, 2006. `http://openafs.org/`.

[3] *Open LDAP*, 2006. `http://www.openldap.org/`.

[4] *CODA Filesystem*, 2007. `http://www.coda.cs.cmu.edu/`.

[5] *Hardened Gentoo 2007.1*, 2007. `http://www.gentoo.org/proj/en/hardened/`.

[6] *Kerberos: The Network Authentication Protocol*, 2007. `http://web.mit.edu/kerberos/www/`.

[7] Achour, Betz, Dovgal, Lopes, Olson, Richter, Seguy, Vrana, et al. *PHP Manual*, 2006. `http://www.php.net/manual/en/`.

[8] MySQL AB. *MySQL 5.1 Reference Manual*, 2006. `http://dev.mysql.com/doc/refman/5.1/en/index.html`.

BIOGRAPHIES

RENÉ HEINZL studied electrical engineering at the Technical University Vienna. He joined the Institute for Microelectronics in November 2003, where he finished his doctoral degree in September 2007. His research interests include computational science, high performance programming techniques, solid modeling, scientific visualization for TCAD.

GEORG MACH was born in Vienna, Austria, in 1979. He studies electrical engineering at the Technical University Vienna. He joined the Institut für Mikroelektronik in June 2003, where he is currently working on his diploma thesis.

PHILIPP SCHWAHA studied electrical engineering at the Technical University Vienna. He joined the Institute for Microelectronics in June 2004, where he is currently working on his doctoral degree. His research activities include circuit and device simulation, device modeling, and software development.

SIEGFRIED SELBERHERR received the doctoral degree in technical sciences from the Technical University Vienna in 1981. Since that time he has been with the Technical University Vienna as professor. Dr. Selberherr has been holding the "venia docendi" on "Computer-Aided Design" since 1984. As of 1988 he has been chair professor of the Institut für Mikroelektronik. From 1998 to 2005 he served as Dean of the "Fakultät für Elektrotechnik und Informationstechnik" at the Technical University Vienna. His current topics of interest are modeling and simulation of problems for microelectronics engineering.