ASEAM

Overview and Specification

Andreas Bauer, baueran@cs.anu.edu.au

The Australian National University
Department of Computer Science
Canberra, ACT 0200

April 17, 2000

Contents

List of Figures

1 Introduction

ASEAM stands for *ANU Student Enrollment And Marks System* used within the Faculty of Engineering and Information Technology. It is designed to store information about student enrollments, marks, unit- and activity information. The ASEAM project involves the creation of such a database system and the implementation of a user-friendly interface to it. In this case, however, it is important that the solution is entirely based upon free software packages which are generally available from the Internet.

People who are meant to use the final product are not only students, but lecturers, tutors and administrative staff. The idea is to provide one easy to use interface for all actors that can be accessed not only inside the department's own facilities, but also from their homes and other locations.

Students need ASEAM to enroll in units, sign up for activities[1] and to view their results in different assignments. Lecturers and tutors want to enter marks for students, handle eventual unit overflows and keep track of all current units and related activities. Once the enrollment period is over, members of staff are able to limit the ASEAM-accessibility of students.

[1] An activity is either the lecture of a unit, a practical group or a tutorial group.

2 Original Situation

Until now the Department of Computer Science (DCS) has been using several different database systems to keep track of unit information [ANU00], student enrollments, marks or assignments. Each component of the employed infrastructure was implemented independently from already existing ones. That is, the authors of these systems used different programming languages and different ways of implementing their database models.

While some parts are written in Tcl/Tk, other components are purely command-line oriented applications that do not warn the user if, for example important data is about to be erased. They also do not offer a user-friendly interface of any kind.

Such an approach is very static and envokes problems not only when trying to exchange information between these database systems, but also when trying to make simple changes to one of the components or when new users have to access and enter data in the system. Furthermore, the administrative effort is very high, considering the fact that each user who wants to access the command line oriented parts needs a separate account on the corresponding server. While some of these "Shell based" parts could at least be used outside of the DCS facilities, the Tcl/Tk parts only work inside the computer labs on campus.

At present the main sources of information and the corresponding applications are ATLES, TYCHO, DAIS and ANUSIS as well as a system for creating time tables and one for viewing student marks.

2.1 ATLES

ATLES is written almost entirely in Tcl/Tk with a web-based user-interface (using Browser Plug-Ins) for administrative staff and a pure Tcl/Tk interface for students. It is used at the beginning of each semester to allow students to register in small-group sessions and to enroll in offered units.

This application does not use a *real* database engine (such as Oracle or PostgreSQL). All relevant information is kept in plain text files stored on the local file system and categorized by directory and filenames. Even though it is not possible to lay out an accurate Entity-Relationship diagram for ATLES [CBS96], Figure 1 illustrates roughly how records within the systems depend on each other. Of course this diagram must lack the essential Primary Keys, as distinction relies mostly upon the names of text files and their directory names. The system can cope with multiple entries for students or units and in fact encourages those. For example a path like

```
/path_to/year_1999/semester_n/unit_description/file_name
```

determines that all files within "unit_description" belong to semester n and to the year 1999.

Also the many-to-many relationships are implemented in an uncommon way: they are not separate tables, storing key values from both partners, but variable

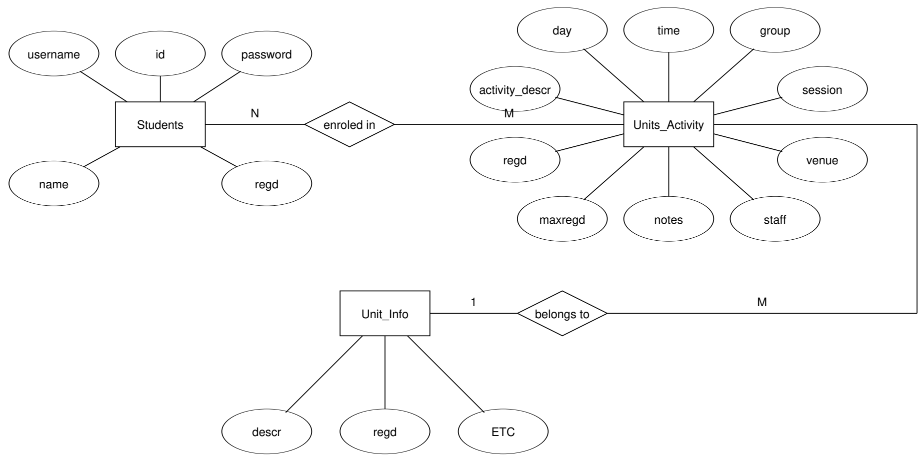

Figure 1: Database model of the ATLES system

text fields, such as "regd" which contains a string of student IDs, each separated by a comma.

The table "Unit_Info" contains unit specific details such as additional comments or names as well as a list of students who are enrolled in a particular unit. The same relationship is also used whenever a certain unit is known by two different descriptions. For example: a unit called COMP110 might be known to students of the Engineering Faculty as ENG110 even though both groups of students would be attending one and the same course. The field "ETC" represents such situations. It indicates that the table "Unit_Info" can be extended with fields, depending on how much additional information has to be stored for such candidates. ATLES is very 'flexible' in that matter.

Further documentation on ATLES can be found at `http://tux.anu.edu.au/Services/Ais/ATLES/`

2.2 TYCHO

Tycho is the marks database system. It is used by academics and part-time tutors during semesters to track student membership in small-group classes, and to store and calculate marks. TYCHO is based upon Shell scripts only.

Even though this component is mainly used for entering student's assignment results it also stores information on the unit's activities, such as the location of each event, date and time, the event's reader, etc. Some of this is helpful to distinguish which tutor has to enter marks for which course - a lot of information is redundant though as ATLES (and also other database systems) already keeps track of similar entries.

Just like ATLES, TYCHO stores in plain text files located in distinctive directories on the local file system, which of course creates even more redundancy than ATLES does. No *real* database engine is implemented which would be using Primary Keys or other essential features [CBS96]. TYCHO's internal structure can be roughly illustrated in a database diagram (whose tables I have put together as a "Unit Catalogue" - see page 7). Such a "Unit Catalogue" con-

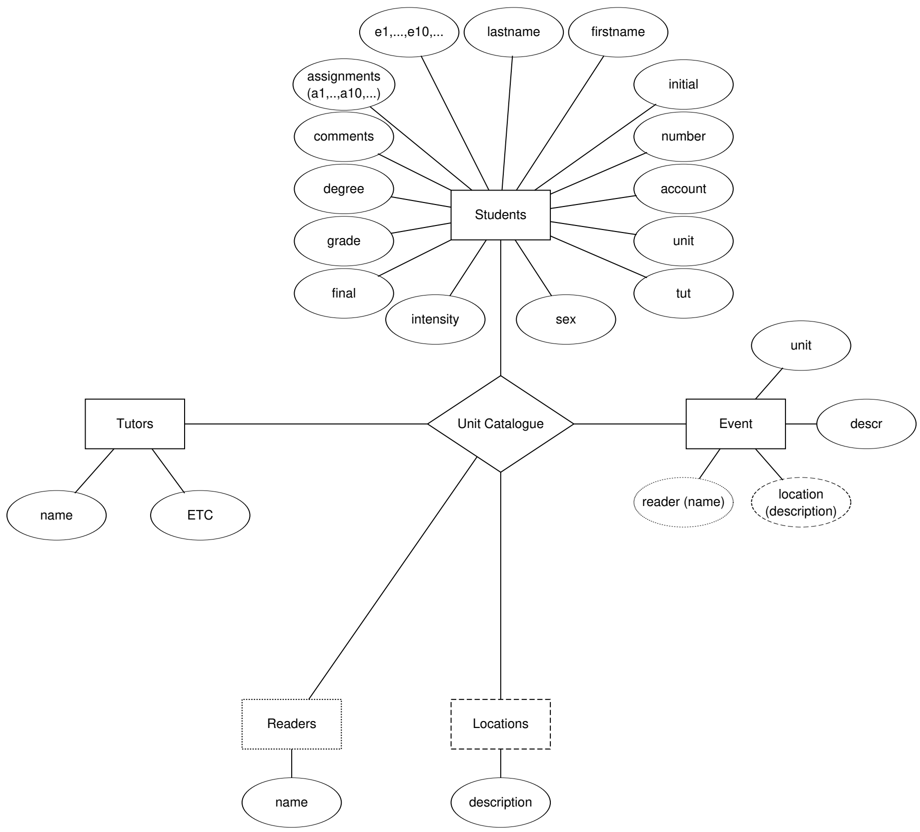

Figure 2: Database model of the TYCHO system

tains all the information of one single unit and consists of the tables "Readers", "Tutors", "Locations", "Students" and the "Events" themselves (lectures, tutorials or practicals). A complete collection of "Unit Catalogues" would represent the University's entire unit offering from the past and present. Some of its structure can also be expressed simply via the directory tree:

```
/path_to/year_1999/semester_n/unit_catalogue/file_names
```

Thus, similar to ATLES, student information gets duplicated with each unit the student has enrolled in - each year. Additionally, all the information on students contained within ATLES is kept in TYCHO as well. The system also stores staff members' names, the events' times and locations etc.

Most important for TYCHO's functionality are the fields e1, e2, ... and assignment1, assignment2, ... These unique fields contain the marks and assignment descriptions for a specific student and unit. If there are two assignments for one unit, the students will have e1, e2 and assignment1 and assignment2. If there are ten assignments for a unit then students will have ten fields each. There is no limit on the number of fields.

Further documentation on TYCHO can be found at `http://tux.anu.edu.au/Software/TYCHO/`.

2.3 Student Marks System

The Marks System is merely a small program allowing students to view results and assignments from within the Computer facilities of the DCS. It is written in Tcl/Tk and some parts in the language C. Its data records are stored in plain text files, extracted from TYCHO via `cron`-jobs.

There is no further documentation available (at least not that I am aware of or that I have been informed about).

2.4 DAIS

The Departmental Administrative Information System. DAIS is used by the DCS staff to track high-level information about personnel, including locations and units, tutors, enrollments and results. Most importantly though DAIS is the main source of unit/activity times and locations for ATLES and TYCHO, calculated by a separate program (see 1.2.6 The Timetable Program). ATLES imports this information from DAIS , TYCHO then imports the information from ATLES. The documentation for DAIS can be found at `http://tux.anu.edu.au/Services/Ais/DAIS/`

2.5 ANUSIS

This is the most comprehensive and important database system used inside the Australian National University. It stores all relevant information about each student on campus and provides other database systems with the names of students, their IDs and further details.

The ANUSIS system is based on Windows NT and offers no external interface to academics- not even to DCS staff. For security (and also political) reasons it remains to the non-administrative staff a "Black Box". Thus a lecturer teaches a class, but is not allowed to directly see who is attending the class. Instead ANUSIS exports two seperate text files containing relevant unit- and student information to be imported into ATLES and other database systems which then are used by DCS academics. Usually these imports are handled by Shell- and Perl scripts, others via "pen and paper".

2.6 The Timetable Program

DAIS and ANUSIS both get information on courses and units from the time table program used on campus. This includes starting and ending times as well as the locations of the activities. More information can be found at `http://timetable.anu.edu.au/`.

2.7 Data Flow

The information flow between all of these databases is generally handled by Shell or Perl scripts (except those "pen and paper" solutions) which run during the night. Records, stored in any of these systems are usually not directly readable by any of the other components, since very individual file formats are used which invoke a lot of converting effort for each single record or entire files. The flow of data between all of the components is depicted on page 10.

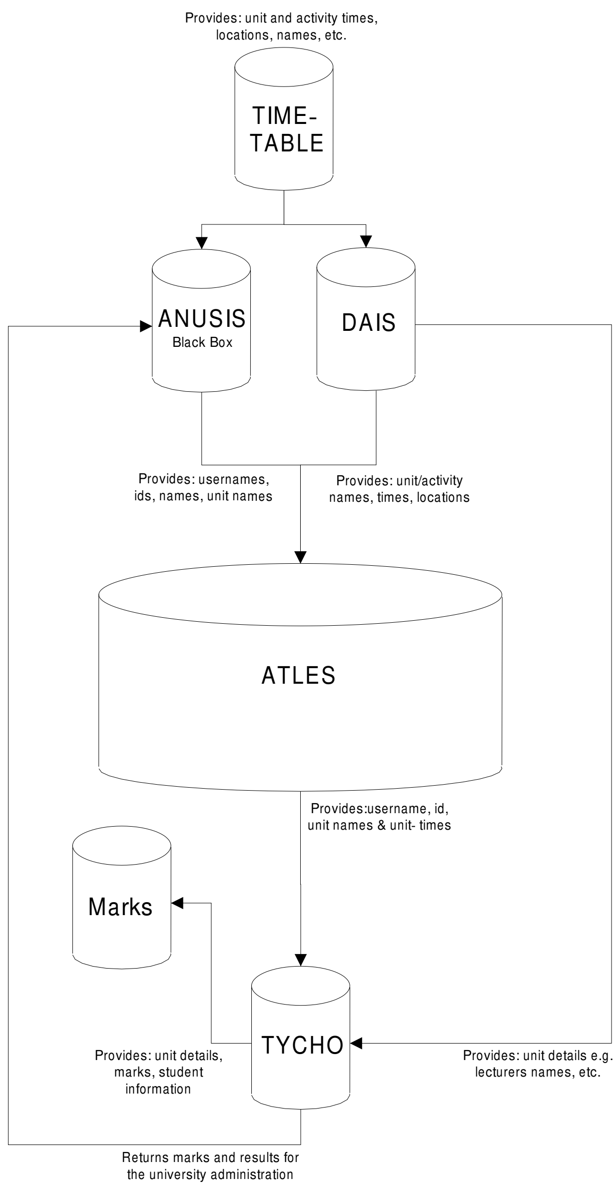

Figure 3: Dataflow between all the database systems

3 Objectives

The following section outlines the fundamental requirements of the ASEAM system, its design goals and security issues.

3.1 Functional Requirements

Most of ASEAM's functionality is derived from ATLES and TYCHO. A key feature of ATLES is to track activity and unit information. TYCHO's key feature is to modify student's marks and their assignments. Thus ASEAM provides these aspects from both systems and interacts with the same group of users.

3.1.1 Main Actors

The ASEAM database system interacts with two actors:

- The student who (un)enrolls in units, signs up for activities, leaves activities and views marks and assignments.
- The member of DCS staff who changes the students activities and courses, keeps track of the students who are enrolled in specific units, enters assignments for a unit, enters the students marks for those assignments, views statistics for unit assignments, and changes unit and activity information such as course limits or descriptions, etc.

Students and DCS staff will authenticate themselves to ASEAM by the information which is stored in the University's central identification database[2]: username and the user's password. The username of DCS staff is also used by the ASEAM identification-database to allow academics access to an improved staff user-interface.

3.1.2 Unit Information

A unit can have many activities. These are divided into Practicals, Tutorials and one Lecture per unit. Each activity is held by a member of the DCS staff, including the lecture. The lecturer usually creates the assignments for a unit and individual tutors usually enter the marks for all students who signed up for their group. This distinction is not necessary though - the lecturer is provided the same functionality as tutors and therefore is allowed to enter marks as well (and vice versa, if necessary).

[2] Until now, Sun's NIS+ authentication system is employed throughout the campus. But as ASEAM was already designed and developed to directly access the NIS+ database, a political decision was made to change that and make use of an LDAP server in the following year [LD00]. All further references to ASEAM authentication issues should be considered from that point of view. However, even in the current state of the ASEAM project all users must provide their valid username along with their password in order to use the ASEAM system.

3.1.3 Enrollment

Students can only (un)enroll in units of the current year and usually only within the enrollment period. By default a student may chose only one Practical and only one Tutorial for each unit he or she is enrolled in. Staff may change these default values.

Each activity, except for the lecture, has a maximum limit on the number of students allowed to participate. If the limit is exceeded students can no longer sign up. Staff may, however, sign up a particular student even though the limit is exceeded.

DCS students can enroll in units which are known to students of the Engineering Department (for example ENG1100 equals COMP1100) by a different name even though they are really one and the same unit, held by one and the same lecturer, offering exactly the same Practicals and Tutorials (and vice versa).

Once the enrollment period has expired the DCS staff may decide to limit the system's accessibility for students. In that case students can only view marks and results, but not change their enrollment status.

3.1.4 Marks

Students are not able to change their marks. They can view them either by unit or obtain a complete list showing all marks that are stored in the ASEAM system. DCS staff enters the marks for each student and each assignment.

3.2 Nonfunctional Requirements

- The system should be based upon free software which is available from the Internet
- The system should not be limited to a certain number of units or students
- The actors are authenticated using a username/password combination
- The system uses secure mechanisms to transmit personal information
- Additional data/fields can be added to the system at any point and at runtime
- Existing data within the system can be exported into other external systems
- The system should backup the state of the database every day such that it can be recovered in case of data loss

3.3 Security

At the moment all users at the DCS authenticate to the University's Computer facilities via a central authentication database system (see also chapter 4.1.3) by using a username/password combination. The ASEAM database system uses the same information to grant access to its actors - even outside of campus. Especially for the latter all significant user details are transmitted to the system using strong data encryption mechanisms.

The server on which ASEAM is implemented on provides no other services (that is, `inetd` must not be started!) than serving web pages. Only authorized clients are allowed to access the web server and the database application. An unprotected public page provides the facility for users to log in (generally the server's main index page).

3.4 Target Environment

- Actors interact with the system using a web browser without support for Java or JavaScript, etc.
- The persistent storage is realized using a freely available SQL database system.
- The business logic is realized using a server-side application.
- Sensitive information gets transmitted encrypted.
- User authentication is handled via the information stored within a central database system.

3.5 Top Level Design

The ASEAM database system includes different subsystems and modules. The main ones are:

- a storage subsystem managing persistent data
- a student user-interface
- a staff user-interface
- a user-interface subsystem translating HTTP requests into requests to other subsystems and translating results back into HTML pages
- a session subsystem controlling basic authentication (for the ASEAM database itself), check in and check out
- a database-connection subsystem controlling a pool of database-connections the users may borrow from (or return to) in order to access ASEAM's database

- a data import and export subsystem
- a separate authentication subsystem to protect the entire web server from unauthorized access

4 Design

This chapter describes the conception of the ASEAM database application, the involved subsystems as well as the software environment to run and modify the system.

4.1 Communication Model and Architecture

4.1.1 Application Scheme

Generally all clients interact with ASEAM via a simple web browser. The ASEAM business logic is realized with the Java Servlet technology which means the web browser does not even have to support Java or JavaScript itself (nor any other kind of plug-in). A Java Servlet receives the client's HTTP requests (`GET`, `PUT`, `POST`, etc.), translates and forwards them to its persistent storage subsystem and returns the result, embedded in HTML tags.

ASEAM uses an SQL-database engine, offering the great advantage of having a standardized and well documented interface that many other programs and programming languages support. A JDBC driver is used for translating user queries as well as results between the Java Servlet and the database engine itself.

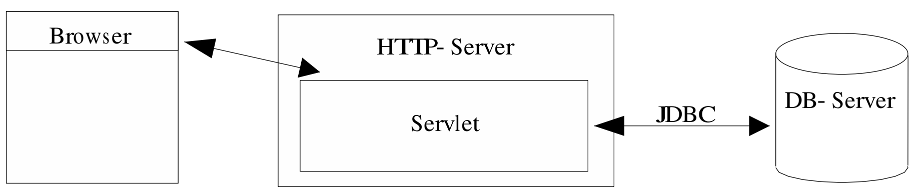

Figure 4: Basic application scheme of ASEAM

Security Even though it appears that the browser sends merely plain HTTP requests to the server, it is not what *really* happens. Except for the web server's main index file, all further communication between browser and server is handled via the HTTP Secure protocol (HTTPS). That is, HTTP over SSL (an additional transport protocol on top of TCP/IP providing communication authentication, privacy and integrity through strong cryptography techniques) and thus a secure HTTP variant [E00]. SSL is short for "Secure Socket Layer".

More information concerning security issues can be found in chapter 4.1.3.

4.1.2 User Interface

All clients interact with the ASEAM system through a web based user interface. The ASEAM Servlet generates a dynamic and user specific HTML output to provide all necessary functions. No further browser-side language support is required.

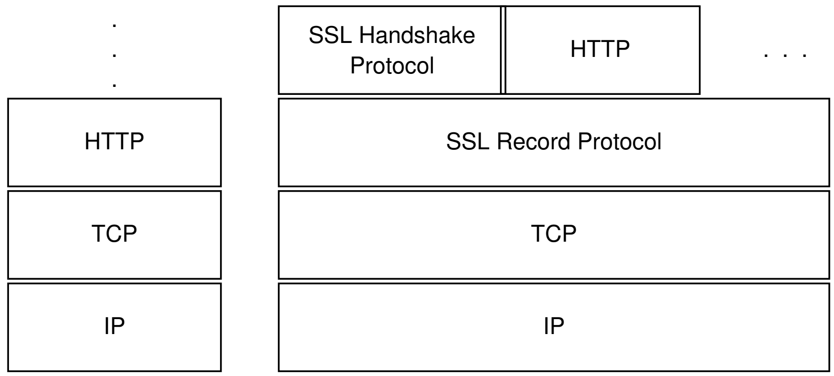

Figure 5: Transport protocols without SSL *(left)*, SSL protocol stack *(right)*

Users request information by clicking on links or buttons in their browser windows which the browser transmits as HTTP requests to the Servlet. The Servlet then reacts to each request (`GET`, `PUT`, `POST`, etc.) and invokes its methods with parameters. Such parameters may be created within the Servlet or transmitted as part of the request. Methods may interact with the data storage subsystem and send back HTTP responses, embedded in HTML tags, for the user's web browser (see also Figure 4).

4.1.3 User Authorization

One of the most important features of ASEAM is the fact that it can be accessed not only inside the ANU computer facilities, but also from students' homes, even students on holiday in Sydney or Belgium. All clients trying to establish a connection to the system have to authenticate themselves with a username/password combination that gets verified by the University's NIS+ system.

NIS+ stands for *Network Information Service Plus* and is used on campus to track usernames, passwords and user rights. It was designed to replace NIS, and is a default naming service for Solaris [NIS00]. It increases security by using an additional authentication method. Enrolled students (as well as staff) have entries in the NIS+ database to access computer facilties on campus.

The web server of ASEAM uses this database to authorize HTTP requests each time they are made. However, the username and password have to be provided only once per session, in order to log in.

4.2 Software Environment

The Software Environment explains which software packages have to be installed in order to run and modify the ASEAM database system. An installation example demonstrates this procedure step by step.

4.2.1 Installation

PostgreSQL A source distribution as well as a precompiled binary distribution of this database is available from the web page `http://www.postgresql.org/`. ASEAM uses the Linux version 6.5.3. Other freely available SQL-database engines may have been a sensitive option (such as mSQL, MySQL), but PostgreSQL is the most powerful implementation when it comes to performance and flexibility issues[3] [SQL00].

Java Database Bridge (JDBC) PostgreSQL provides its own Java driver to access data from Applets or Servlets. Each version of the database engine needs its own unique version of the driver. ASEAM uses the precompiled JDBC 6.5-1.2.

Java Development Kit (JDK) Only JDK 1.2 (or higher) will work with the JDBC driver of PostgreSQL 6.5.3. The Linux port of the JDK is available from Blackdown's web site: `http://www.blackdown.org/java-linux.html` (also `http://java.sun.com/`). It is not recommended to use "Kaffe" whatsoever (as it is provided with various Linux distributions such as Red Hat for example).

Java Servlet Development Kit (JSDK) By the time this documentation is written, the latest version of this tool is 2.1. ASEAM employs the old version 2.0, because the ApacheJServ module (current version 1.1) only works in combination with this particular release [ApaJ00]. As with all of the other programs, JSDK can be downloaded from the world wide web: `http://java.sun.com/`.

Apache web server Apache 1.3.12 is used as Http-server, whose distribution is freely available from the Apache web page: `http://www.apache.org/`. It is recommended to get the source-distribution and to wait with a proper installation until the Apache JServ- and the mod_ssl modules are fully compiled. The example session (as well as the documents provided with Apache JServ [ApaJ00]) explains this more in detail.

Although a different server could have been used (e.g. one that is entirely written in Java, like Sun's own server) Apache sofware was chosen because it is free and has been used and administered in the Department of Computer Science for years.

[3] According to the MySQL documentation [My00], PostgreSQL is a lot slower in performing `INSERT`s or `UPDATE`s. However, PostgreSQL is faster in highly complex queries (`JOIN`s) due to its extensive use of index files when building the cartesian product of tables. MySQL uses a huge temporary file instead which naturally depends on hard disk- and controller performance, etc. On top of that it is not even possible for MySQL to perform as complex queries because various sub selection criteria are not yet supported.

Apache JServ This package, in the current version 1.1, can also be downloaded from one of the Apache sites: `http://java.apache.org/`. Once the main Apache web server is configured, JServ has to be configured and compiled. The example session explains this a bit more in detail.

OpenSSL OpenSSL 0.9.5 is a free encryption software, based on SSLLeay, available from `http://www.openssl.org/`. It has to be configured and compiled after JServ was successfully built (see example).

mod_ssl Just like JServ, mod_ssl 2.5.0 is an extension to the Apache web server which also needs OpenSSL in order to function. A source distribution is available on its web site: `http://www.modssl.org/`. It has to be configured after OpenSSL builds (see example).

4.2.2 Example

This is a step-by-step example session, showing how the installation process could look alike on a system. In this example we have already installed PostgreSQL (and started the server demon with option -i), the Java Development Kit (JDK1.2.2) and the Java Servlet Development Kit 2.0 (JSDK2.0). All the remaining packages have been downloaded and unpacked successfully.

- Now change the directory to the Apache source distribution and run `configure`:

  ```
  ./configure --prefix=/usr/local/apache
  ```

- Change the directory to the ApacheJServ distribution and run `configure`:

  ```
  ./configure --with-apache-src=/tmp/apache_1.3.12
  --prefix=/usr/local/jserv
  --with-jdk-home=/usr/local/jdk1.2.2
  --with-JSDK=/usr/local/JSDK2.0
  ```

- Compile the ApacheJServ module by running `make` and then `make install`:

  ```
  make
  make install
  ```

- Change the directory to the OpenSSL distribution and run `config`:

  ```
  ./config
  ```

- Compile OpenSSL running `make`:

  ```
  make
  ```

- Change the directory to the mod_ssl distribution and run `configure`:

```
./configure --with-apache=../apache_1.3.12
--with-ssl=../openssl-0.9.5
--prefix=/usr/local/apache
--activate-module=src/modules/jserv/libjserv.a
```

- Change back to the Apache directory and compile by running `make`:

  ```
  make
  ```

- Now create a certificate for your secure web server and answer all the following questions of `make certificate` carefully. The most important question though, whether this is only a test certificate or not, is No. 6 ("Common Name"). The server name must be the name of the machine you want to certificate, not the main server of the department or anything else, since mod_ssl uses this value to access the web server.

  ```
  make certificate
  ```

- Create the main apache-directory by running `make install` in the Apache distribution directory:

  ```
  make install
  ```

Automating the Installation A Shell script to automate the installation task can be found on this web site: `http://www.in.tum.de/~baueran/aseam/installation.sh`

Configuration files It is also necessary to spend some time configuring the settings for each module and program before running applications or starting to change the Java Servlets for this environment.

The Apache configuration file `httpd.config` should be considered first. Following, a list of possible changes to make:

- Advise Apache to use a lock-file:

  ```
  LockFile /usr/local/apache/logs/httpd.lock
  ```

- Configure the web master's e-mail address:

  ```
  ServerAdmin web master@cs.anu.edu.au
  ```

- The server name:

  ```
  ServerName vega.anu.edu.au
  ```

- Avoid accessibility to any directories, but those needed to run the Servlet application:

```
<Directory />
  Options FollowSymLinks
  AllowOverride None
  Order deny,allow
  Deny from all
</Directory>


<Directory /usr/local/apache/htdocs/>
  Options FollowSymLinks
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>


<Directory /usr/local/apache/htdocs/application/>
  AuthType Basic
  AuthName "The ASEAM System"
  AuthUserFile /home/aseam/passwords
  Options FollowSymLinks
  AllowOverride None
  Order deny,allow
  require valid-user
</Directory>


<Location /servlets/>
  AuthType Basic
  AuthName "The ASEAM System"
  AuthUserFile /passwords
  AllowOverride None
  Order deny,allow
  require valid-user
</Location>


<Location /servlet/>
  AuthType Basic
  AuthName "The ASEAM System"
  AuthUserFile /passwords
  AllowOverride None
  Order deny,allow
  require valid-user
</Location>
```

- For our purpose it is also recommended to switch off the possibility for users to run any CGI-scripts:

  ```
  <Directory "/tmp/apache/cgi-bin">
    AllowOverride None
    Options None
    Order deny,allow
    Deny from all
  </Directory>
  ```

- Make sure the port configuration is correct. The standard port should be 80, the SSL-capable port should be 443. In order to have port numbers $<$ 1024 you must start the server as user root. However, if you define User and Group correctly, the server-demon-task changes its ownership back to these settings:

  ```
  User baueran
  Group dcsstaff
  ```

- And finally, include the support for the JServ module by adding this line to the very bottom of httpd.config:

  ```
  Include /usr/local/jserv/etc/jserv.conf
  ```

The next file which should be modified is JServ's `jserv.conf`. Here's a list of possible changes to make:

- Set the default host:

  ```
  ApJServDefaultHost localhost
  ```

- Make sure to disable the secret key option:

  ```
  ApJServSecretKey DISABLED
  ```

- Now you should configure the Servlet zones. Take a minute to read the comments in the configuration file itself. Normally you do not need to change anything at this point, though you might want to do so later.

The last file that needs to be modified is the `jserv.properties` configuration file.

- Tell the Servlet engine where to look for the database bridge file (JDBC):

  ```
  wrapper.classpath=/usr/lib/pgsql/jdbc6.5-1.2.jar
  ```

- Make sure the right protocol is set:

  ```
  wrapper.protocol=ajpv12
  ```

- At this point you might eventually have to configure the Servlet zone's properties. Usually this is not to be done though from the beginning.

- Tell the Servlet engine which addresses may access:

  ```
  security.allowedAddresses=127.0.0.1, 150.203.160.27
  ```

Symbolic links If no changes were performed within the Servlet zones, all Java Class-files have to be placed in the servlets directory of the JServ package. It is recommended to put the actual Class files outside the directory tree which is accessible by the web server and to create merely symbolic links to them.

Environment variables The `CLASSPATH` has to be set to the actual JDBC driver location, the path to the Servlet classes, to the JSDK classes and to the Servlet engine itself. A properly set path could look like this:

```
CLASSPATH=/usr/lib/pgsql/jdbc6.5-1.2.jar:
/home/aseam/src:
/usr/local/JSDK2.0/lib/jsdk.jar:
/usr/local/jserv/libexec/ApacheJServ.jar
```

The `JAVA_HOME` varibale gets usually set during the installation process:

```
JAVA_HOME=/usr/local/jdk
```

Starting the server By default all Java Servlets belong to `jserv/servlets/` (if no other Servlet zones were added). So make sure all the essential Class-files of the Servlet application are copied (or linked) to this directory. The according HTML-files should be placed within the htdocs directory.

Now you should make yourself known to your system as `root` and start the server in the `apache/bin/` directory:

```
./apachectl startssl
```

4.2.3 Server Certificates

The above example covers most aspects of the software installation. However, in order to start a web server with SSL support it is necessary to receive a digital server certificate from an issuing authority such as *Thawte Digital Certificate Services* (`http://www.thawte.com/`). Each company provides its own guidelines on how to receive such a certificate which explains why this step is not covered in this documentation.

In general the web master has to send proof of its company's identity, then generates a temporary key (similar to the procedure of `make certificate`) and receives the registered personal certificate from the issuing authority a week later. A basic certificate costs about hundred Dollars, an enterprise solution up to several thousand Dollars depending on the ammount of web servers to authorize, encryption methods, etc.

Digital certificates are necessary to confirm to clients who send sensitive information over the web that their data is only read by the intended receipent.

5 Schedule

Task:	**Date:**
Examination of existing DBMS and visualizing the data flow and models	15. Nov. - 21. Nov.
Specification of a new DBMS (ASEAM) replacing ATLES and TYCHO; Creation of a new database model for ASEAM	22. Nov. - 01. Dec.
Comparison of different software packages that could be used for the ASEAM project; Setting up a software (development) environment	02. Dec. - 07. Dec.
Implementation of the new database model in PostgreSQL	08. Dec. - 08. Dec.
Development of a Java Servlet, providing function-anility and user interface for Students	09. Dec. - 15. Jan.
Creation and implementation of a security model for the ASEAM system and especially the web server	16. Jan. - 01. Feb.
Development and implementation of functionality for staff and academics	02. Feb. - 28. Mar.

6 Conclusion

Even though the ASEAM project meets the criteria of this specification quite precisely, some issues had to be modified during the programming process, even shortly before the final deadline. That is, the security model is still unresolved, whether it should rely on NIS+ or LDAP [LD00]. The reasons for this are merely political as the academic staff has yet to decide which system will be used on campus from the beginning of next year (most likely LDAP though). From the developer's point of view it does not matter too much which technique is employed as long as the requirements are clear. However, ASEAM still falls back to the Apache mechanism of user identification (mod_auth) accessing an encrypted file, located on the web server's file system. Basically ASEAM can be modified at a later stage to support any new desired technique.

All other topics discussed in this document have more or less derived from various discussions with University staff and by trying to work out how the old database systems could be improved. By the time the ASEAM project was considered first, there was no list of requirements or objectives available.

References

[ANU00] Australian National University. Administrative Information Systems. http://tux.anu.edu.au/Services/Ais/index.html, 2000

[Apa00] Apache Software Foundation. Apache 1.3 User's Guide. http://www.apache.org/docs/index.html, 2000

[ApaJ00] Apache Software Foundation. Java Apache Project. http://java.apache.org/index.html, 2000

[CBS96] Thomas M. Connolly, Carolyn E. Begg, Anne D. Strachan (1996): *Database systems: a practical approach to design, Implementation, and management*, Addison-Wesley

[Dav98] Thomas Davis (1998): *Build your own object pool.* JavaWorld Magazine, June 1998

[EN94] Ramez Elmasri, Shamkant B. Navathe (1994): *Fundamentals of database systems*, Benjamin/Cummings

[E00] Ralf Engelschall. modSSL - *The Apache Interface to OpenSSL.* http://www.modssl.org/index.html, 2000

[HC98] Jason Hunter, William Crawford (1998): *Java Servlet Programming*, O'Reilley

[My00] MySQL. Documentation. http://www.mysql.org/docs.html, 2000

[LD00] OpenLDAP. Documentation. http://www.openldap.org/index.html, 2000

[SQL00] PostgreSQL. Documentation. http://www.postgresql.org/docs/index.html, 2000

[NIS00] Solaris NIS+ Frequently Asked Questions And Answers. http://www.eng.auburn.edu/users/rayh/solaris/NIS+_FAQ.html, 2000

[Sun00] Sun. Java Documentation. http://java.sun.com/docs/index.html, 2000