A SIP-enabled communication portal for e-learning

Final Project Report version 0.9

from:

Markus Beier, Caroline Daniel, Daniel Dudaszek, Marc Hermann, Stéphanie Lecat, Daniel Toeller

Contents

Chapter 1

Introduction

1.1 Abstract

The goal of this project was to build up an infrastructure enabling tele-tutoring services. The departing point for the project is a scenario where a university for example wants to offer tutoring services to a group of students using the new possibilities of the internet. Until now the usual way of offering tutoring at universities is to offer the services of a tutor at certain time frames at a certain location. In the raising field of eLearning, these offers are not so easily providable. With learning not via traditional lectures but via electronically offered content, like the CBT of the University of Mannheim or via learning applets, the way of learning changes. The students now can work with these things wherever and whenever they want. So you don't have a homogeneous group of students which all should have heard the same part of the lecture, but a heterogeneous group with very different levels of knowledge. The physical distribution of the participant is an additional factor impeding traditional tutoring services. This leads to new forms of tutoring which adopt these facts. Our task was to install an infrastructure enabling the provision of these new tutoring services.

This field is closely related to the traditional customer support and customer relationship management. The customers got the product and questions on this product arise at different locations and times so a physical meeting of the supporter and the customer is often difficult and expensive. But in most of the cases physical presence is not necessary, which led to the development of the current call-centers. Because of this relation we researched how business customer relationship management is done via elaborate software products on the example of SAP CRM 3.0 in addition to other techniques.

We implemented a Web Site working as a portal for the requesting students. There they can find out how to contact the tutors. The communication will be established via SIP enabled phones. At the end of the report an outlook is given how the possibilities SIP offers could be augmented in the future.

1.2 Problem Description

The problem for which we shall provide a solution is that a certain number of students can contact a certain number of tutors at any time from anywhere, no matter what time and what place.
Every student knows the problems of reaching the tutor exactly at this or that time. Sometimes questions arise not only at the tutorium - then they have to be answered and for this the tutor has to be reached.

The problems arise at different times and different locations so their solution should be available at any time and anywhere. For example the tutor should be called on his telephone if he is at home, on his business phone if he is at work and on his mobile phone if he is on the way from his home to his work. The requirement at any time is quite impossible to solve in the environment of

a university due to budget limits (there has to be a tutoring service around the clock - someone should always be able to pick up the phone or answer an email), but the initiation of contact should be more flexible and not limited to a time-frame of for example three sessions of 90 minutes like it is today with the tutorium lessons. This impedes solutions like traditional tutorials. Our solution is one of the first steps to establish a more personalized way of learning than it was before. It doesn't replace the traditional tutoriums - it enhances them.

The communication should be possible at least via voice, because this allows the student to specify his questions in a more exact way and many questions only arise, when you are in a talk.

The desired level of contact possibilities include videoconferencing, shared files and shared whiteboards. Whiteboards can be very useful if the tutor wants to demonstrate complex facts, draw figures or show animations. It is more efficient than just answering email-questions.

1.3 Internet communication services

The task of the project is to provide a Web-based approach. Therefore we give here an overview of the available technology.

1.3.1 eMail

The usage of eMail to get in contact with a tutor is already quite common. Normally the tutor gives his eMail address to his students to offer a platform for the solution of problems which may arise at home when the student review the lecture, their notes or prepare the coming lectures. It is easily realizable because every student has an eMail address through the university. Almost all students have access to the internet from home. Those who don't have a computer at home can use the infrastructure of the university.

This already classical communication instrument is integrated in our platform. No special integration was necessary because all current browsers support the basic mailto: command. So regular hyperlinks are provided.

1.3.2 Web Portal

We decided to have a Web Portal to offer the functionality of the contact base. There, all basic information can be stored like the contact data of the students and the tutors. If problems arise, the web portal serves as a departing point for the solution. To store the user specific data, a MySQL database is used and to provide the dynamic creation of the webpages they are implemented in PHP3.

1.3.3 Voice over IP

A relatively new component is the field of voice and video communication over the internet. The communication via eMail lacks of a certain personality. And the speed of the communication flow is extremely slow. If the student asks a question and doesn't formulate it well, the tutor might have to ask for more details. Assuming that both parties check and answer their mail only 3 or 4 times a day, the solution of a small problem could last over a day. In our experience the lead time of a request is more than 12 hours.

Here lies the primary advantage of the telephone services. Internet telephony can provide a cheaper solution of the mapping problem mentioned above. Even video communication could be established. The constraint of these solutions is the connection. At low bandwidth the quality of the link is poorer than via a regular telephone but it is still acceptable. For video communication there must be a higher bandwidth.

The hardware needed consists of a computer with soundcard, speakers, a microphone (alternatively a headset) and an internet connection. Virtually all of the students owning a computer fulfill these requirements. If not they are available at very low rates (most of the students would only

need a microphone in addition to the already present equipment). Alternatively hardware VoIP Phones can be used like the Pingtel xPressa. These phones already include the needed software.

The software requirement for computer users consists of a softphone.

VoIP technology is divided in two major parts. A signalling part and a media streaming part. Signalling means the allocation of the desired callee and the negotiation of the parameters of media transport, while the media streaming describes the protocols used for coding/decoding of the media data and their transmission via IP. In the field of signalling there are two major protocols. One is the quite new Session Initiation Protocol (SIP) and the other one is the older H.323 standard. The media streaming protocol used is the Real-time Transport Protocol (RTP). For our project we have chosen to implement a SIP infrastructure, because it is easily augmentable in its functionality due to the clear separation of the signalling and the media part.

Chapter 2

The Web Portal - Design and Implementation

2.1 Desired Functionality of the Web Page

First, to access the web page it is necessary to log-in. We need to be sure that even if someone knows the address of a specified part of the web page (ex : the page groups.php3) if he has not logged-in, the log-in page will appear instead of the web page.

Within the web site it is possible to email, call via SIP or Netmeeting every person that is displayed.

2.1.1 Student

If a student wants to create an account he has to give a login name, a password, then he should fill out a form with his name, first name, address, e-mail, SIP address, phone number... This information will be stored in the database.

When you log-in as a student, first you can see if a teacher has left a message, then you can see the courses you're allowed to join this semester and have information about them, join or quit a course. You can also see the courses you have already joined. It is possible to check the work group you are in, to join a group and to create work groups. You can also have the list of the students that are in the same courses like you, and the teachers of these courses. And last you have the possibility to edit your profile.

2.1.2 Teacher

If a teacher wants to create an account he has to ask the administrator. Almost the same data like in the case of a student will be asked.

When you log-in as a teacher you can see the courses you are in charge of. You can add, delete or modify your own courses. You have the possibility to list the students in your courses. And you can see the work groups you're in, join and add a work group. And last you have the possibility to edit your profile.

2.1.3 Administrator

When you log-in as an administrator you can see everything about the courses, groups, students and teachers.

You can delete any workgroup.

You can also add a teacher and create his account.

2.2 Technologies used

2.2.1 The MySQL Database

The MySQL software provides a SQL database server.

The MySQL software has Dual licensing, you can use MySQL Server free of charge under the GNU GENERAL PUBLIC LICENSE . You can also purchase commercial MySQL Server licenses from MySQL AB if you do not wish to be bound by the terms of the GPL.

The MySQL database server embodies an ingenious software architecture that maximizes speed and customizability. Extensive reuse of pieces of code within the software and an ambition to produce minimalist but functionally rich features have resulted in a database management system unmatched in speed, compactness, stability and ease of deployment. The unique separation of the core server from the table handler makes it possible to run MySQL under strict transaction control or with ultra fast transactionless disk access, whichever is most appropriate for the situation.

Also MySQL is very easy to use with PHP the language of the web site.

One constraint of MySQL is that one cannot make nested select queries. In conclusion they have to be simulated by hierarchical select queries.

2.2.2 phpMyAdmin

phpMyAdmin is intended to handle the administration of MySQL over the web.

One of the good point of the MySQL interface (phpMyAdmin) is that even if you have never done any MySQL's query you'll have no problem to administrate the database.

2.2.3 PHP3

PHP is a script language, embedded in a normal page HTML, executed by the Web server. It is designed to create dynamic web pages quickly. The result of the program is sent to the software of the visitor of the site, in the form desired by the programmer, normally a standard page HTML. It is thus possible to use PHP to create web pages, with consist of external information (data bases, files, other external pages, sites) or of calculations (display information related to the visitors preferences, the date or various data). PHP is a relatively simple language, but it however contains a great number of functions which make it capable to solve all kind of problems. PHP is one of the language most used on Internet.

The advantage of the use of PHP The PHP Code is executed only by the server, it does not remain trace of the program in the page received by the visitor, only the result. The advantage of PHP compared to other languages like Java or Javascript, is this execution by the server which does not require special software on the computer of the visitor. It is thus 100% compatible with all kind software which is capable to display HTML pages.This is the reason what makes it the language of choice for the creation of WAP sites which can be displayed by very simple browsers e.g. in mobile phones.

How a program PHP arises The PHP code is embedded in a standard HTML page, using special tags beginning with <?php and ?> at the end .

Example:

```
< HTML >
    < head><title>Example</title></head >
    < body >
        <?php echo "Text printed by the program"; ? >
    </body >
</HTML >
```

At the end of its execution the part which goes from <?php to ? > is removed or replaced by the result of the program.

Examples of the possibilities of PHP A very brief list of the various possibilities offered by PHP:

- Management and use of data bases, like MySQL .
- Use of data entered by the visitors (directories, forums, commands,).
- Assembly of pages or files even if they are stored on different computers.
- Adaptation of the contents to the visitors.
- specialized Statistics.
- Automatic generating of e-mail (confirmation of form, mailing list, etc).
- Realization of data in all the formats of Internet (HTML, Text, WAP, GIF, etc).
- Execution of other programs (PHP, CGI, Perl, etc)
- Interaction with other script languages like (Javascript, WMLScript, etc)
- Use of many functions of calculation and handling of texts or images.

The principal assets of PHP are:

- exemption from payment and the availability of the source code (PHP is distributed under licence GNU LPG)
- the simplicity of writing of scripts
- the possibility of including script PHP within a page HTML (contrary to CGI scripts, for which it is necessary to write lines of code to display each line in HTML)
- the simplicity of interaction with data bases (many DBMS are supported, but the most used with this language is MySQL, a free DBMS available on the different Unix platforms, Linux, and Windows).
- Integration within many Web servers (Apache, Microsoft IIS...)

2.2.4 PHPLib

Personlized Webpages

When we consider the functionalities of the Tutoring.net webpage we see that the displayed content is very related to the user which connects to the site.

If we observe two different students connecting to the website and they both want to see the courses they are signed in and not a general information on all courses, we see that this content must be generated dynamically for each user. As we already mentioned PHP is very suitable to generate dynamic web content but when we want to generate this content in a personalised way we encounter some problems.

Sessions

The main problem which is not solvable in a trivial way is the HTTP Protocol which is stateless and works using the request response principle. This problem can be solved by using Sessions to create a definite state between server and client.

This leads us to a second problem, how can a server identify in a unique way a client?
One way would be to use the client's IP Adress, but this is unsuitable if we think about techniques which hide the client's IP address like proxies.

An appropriate way to solve these problems is to generate an unique Session ID which can be done easily by PHP using the MD5 hash algorithm like:

```
md5(uniqid("foo"));
```

This Function generates an hash value out of 2^{128} possibilities.

The next issue is how to pass this value between client and server. This is possible in several ways but the most commonly used ways are to pass the value using the GET Parameter or to store it in an Cookie.

At serverside we have to store all the information which belongs to each session. To do this we can use the potentials of the server system, Session data can be stored in a database, in the file system or only in shared memory if we don't need to serialize it for later use.

One of the Basic properties of an Session is its lifetime which might be hours or days corresponding to the web application.

So Sessions help extend stateless HTTP to a protocol with defined state between the client and the webserver.

Authentication

When a client passes Information to a server which allows the server thereupon to identify the client in an unique way we talk of authentication. To establish this authenticated connection between a server and the client two ways are possible:

1. HTTP Basic Authentication : HTTP Basic authentication can be implemented easily with a few lines of code like this:

   ```
   Header("HTTP/1.0 401 Unauthorized");
   Header("WWW-authenticate: basic realm=\"Bitte autentifizieren\"");

   print "Benutzer $PHP_AUTH_USER\n";
   print "Passwort $PHP_AUTH_PW\n";
   ```

   The code above will result in a pop-up window as shown in figure 2.1.

   It is easy to use but causes some problems:

   - no automatic logout possible
   - no web design possible
   - the user might be lead to uncertainty

2. Session Based Apporach: This Approach uses a Session as already described and offers the possibility to create a webdesigned login screen as depicted in figure 2.2 with an automatic logout function by setting a session lifetime.

   One main disadvantage of this approach is that the whole Session has to be implemented using a real programming language which causes a huge amount of programming Session related necessities before implementing the basic features of the webpage.

Figure 2.1: Login screen for basic authentication

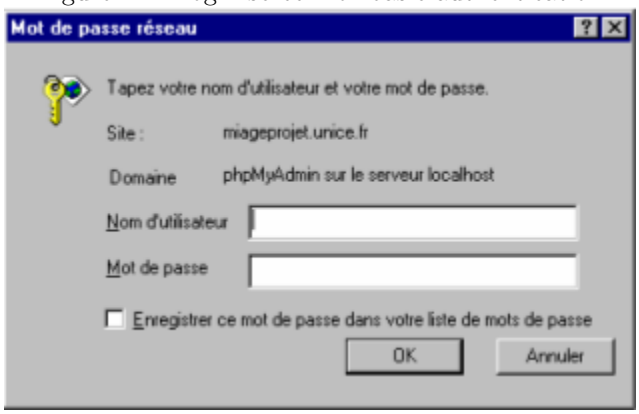


Figure 2.2: Web designed login screen

PHPLib

As already described the creation of a personalised webpages requires a lot of programming to implement all the session and authentication functionalities. As we wanted to concentrate on the main features of the Tutoring.net webpage we decided to use a use a already coded PHP Framework, PHPLIB.

PHPLIB is distributed under the **LGPL** (GNU Lesser General Public License) which means that it can be used free of charges.

Basic Features: The PHPLIB offers as Basic Features:

- Sessions
- Authentification
- Permission Management
- Templates
- Form Layout
- Menu Layout
- SQL Query Tables

Theses Features are implemented as PHP3 Classes, which can be extended to fit the demand of the particular assignment.

The Main Design of PHPLIB

These are the three main classes which have the following main functionalities:

Session: As Sessions are used to store variables for a certain time this class implements all functionalities like the management of Session Ids, the serialisation of variables and the garbage collection of old Sessions.

CT_SQL: This is a helper class which tells the Session Class where and how to store the Session Data. The CT in the name of the class is an abbreviation for Container because this class implements a Storage Container which means it contains all memory access code for the Session Class. If another Container class is used in Session all Session Data will be stored using other Storage Devices than a SQL Database. As this is the SQL Container Class, this class contains a set of SQL queries to load and store Session Variables. It's also used to do the User authentication.

DB_SQL: This class permit access to Databases in a way which is independent from the kind of Database which is used. This is done by various class versions. To give an example db_mysql.inc defines the DB_SQL class for MySQL Databases while db_odbc.inc defines one for ODBC databases. The Basic implementation of DB_SQL is coded generally, it doesn't contain any connect information and only a few defaults to manage the behaviour in error cases.

After this short introduction of main PHPLIB classes, now a short overview about other PHPLIB classes which are also used by our implementation.

Auth: this class is used to create pages which require a login. Every subclass of auth must at least contain the functions auth_loginform() and auth_validatelogin(). auth_loginform() is used to display the Login Screen auth_validatelogin() is used to validate the login, it may return false or the User ID. How this is done is totally independent from the basic Auth implementation.

Perm: This class defines the access rights and different user levels the web application knows. It's also used to print the error message if there are access right violations.

User: Since PHPLib uses USER-ID instead of usernames to identify users it was possible to implement a class user which has the same behaviour as a Session Class but uses User-Ids instead of Sessions Ids. This gives the possibility to register variables directly to users so that this feature of PHPLIB can be used to store personalised settings.

As we have seen many of the abstract functionalities of PHPLIB, we will focus now on the part how to deploy these possibilities on our project.

Installation: As an basic requirement PHPLIB needs a running webserver with installed PHP Interpreter, which was no constraint for our project as a Apache server with running PHP was contributed by the seminar's Administration. The basic idea for customizing PHPLIB is to create subclasses of the already described mainclasses. These classes and their functions are defined in the configuration file local.inc. How this was done in the Project will be described later on as there is also some configuration information needed to link these subclasses. All other configuration is done in prepend.php3

Chapter 3

The Communication Part - A SIP Infrastructure

3.1 Reasons For The Choice of SIP

We have chosen to solve the problem using a SIP based infrastructure because we think it fits best to the requirements. The scalability is given. A great advantage is its modularity. That means that the implementation of the infrastructure can be realized in several steps. First a PC to PC voice communication can be established. After that step is completed. The very same infrastructure can be easily extended to provide PC to Phone, Phone to PC or Phone to Phone calls. The reason for this is the fact that SIP is clearly limited to the signalling. Abstracting you could say SIP offers the possibility to establish a SESSION (like the name already implies) of every kind between caller and callee. The kind of session is only limited to the possibilities offered by the user agent used. So further developments in this sector are almost automatically implemented in the existing structure. New payloads can be defined almost arbitrarily. In consequence every newly developed user agent which implements a kind of session model can use the protocol for establishment of the session. No special directory services have to be used. So SIP enables a certain convergence of session initiation of all kinds.

The great disadvantage of SIP is that it is a very young protocol. It hasn't already made a break-through so there is not yet a huge online community developing. It is more like a widely accepted background technology. For example all SIP servers available are pretty expensive (from $9000 onwards) but therefore high performant. Their target group are ISPs and Telcos who want to offer such services. But there is nothing available like the Tomcat or the Apache in case of Servlets or HTTP Servers respectively. The osr project which tries to offer such free software has just begun so that it is not yet fully functional. But nevertheless the future lies in this protocol. Best example is the Windows XP Messenger, which is included in the Windows XP distribution, uses SIP. The next version of Netmeeting will also switch from H.323 to SIP.

First we give a rough overview over the protocol, after that we describe the infrastructural elements chosen by us. At last we evaluate them regarding some topics like interoperability and scalability.

3.2 The Session Initiation Protocol (SIP)

SIP is an end-to-end, client-server session signaling protocol developed by the IETF. In contrast to H.323 SIP is clearly limited to the signalling part. It does not transport any data. The media streams are transmitted via a peer-to-peer connection between the user agents.

Its standard is defined in the RFC 2543. The SIP infrastructure consists of the following parts :

SIP Registrar The SIP Users are registering on the Registrar Server their current IP.

SIP Proxy Server The SIP Proxy Server receives call requests, tries to locate the callee (via the **Registrar** or DNS) and forwards the request to the given address. This can be done stateful of stateless. Stateful means that the Server keeps track of the session status (for example to resend not answered requests automatically). Stateless Servers are much faster than the stateful ones. The Proxy only takes part at the signalling process but not at the media transmission. The forwarding process can be provided with programmable routing functionalities. Forking is possible as well, that means that if there is an incoming request, this request is forwarded to several sip addresses. The first one accepting the request establishes the connection. To provide scalability Proxy chaining is possible. That means that one Proxy doesn't keep track of all the users but knows to which Proxy Server the request has to be forwarded. This server in the second row provides the regular Proxy functionality.

SIP Redirect Server The Redirect Server acts in the same way as the Proxy regarding the callee location. After locating the callee, it does not forward the call but sends the result to the requesting user agent.

SIP User Agent The User Agent is either a hard - or a software terminal to initiate sessions or receive session invitations. There are many different implementations, such as (Instant) xPressa, EZPhone or Ubiquity Softphone.

The SIP protocol consists basically of 6 different messages to request action from the server. They are called Request Methods like in the HTTP protocol:

INVITE This message is used to invite another user to a session. The session description is given in the body of the message.

ACK This message confirms session establishment.

BYE This message is used to terminate a session.

CANCEL This cancels a pending INVITE message.

OPTIONS With the OPTIONS message the capabilities of the user agent of another user or of a Server can be checked

REGISTER This message is used to update the current address of a user on a Registrar. As the payload of a REGISTER message a CPL script can be uploaded to the Server providing actual call processing logic.

Additionally to the 6 Request Methods, there are also SIP Response Codes, which contain detailed informations regarding the result of the request. Like the Request Methods they are derived from the HTTP protocol

They are separated into 6 groups of information :

- 1.. - Informational, such as forwarding a call
- 2.. - Success
- 3.. - Redirection
- 4.. - Client error
- 5.. - Server failure
- 6.. - Global failure

SIP Message structure:

It consists of a header containing information about the callees address, the callers address and a unique call-id. The Payload contains the information about the session to be initiated, usually the Session Description Protocol SDP is used to inform about the media to use, the session name and contact informations.

In case of a REGISTER message it can be the actual CPL Script of the user.

For media streaming RTP is used just like in H.323.

The SIP infrastructure can be connected to PSTN networks using SIP/PSTN-gateways which consists of sound processor converting the digital media data to analog data and a SIP interpreter which receives SIP messages converting the called number into dial tones.

3.3 Call logic

This section describes the different possibilities the SIP infrastructure offers to put call logic into the processing of the call requests.

3.3.1 SIP-CGI

CGI is the short form of Common Gateway Interface. It enables websites to interact with databases and other programs. For example it is possible to create an application with CGI which regulates the product selection and ordering process in an online purchase. So CGI make the interaction between user and web-server possible.

The CGI scripts can be written in current programming languages like Perl, TCL(Tool Command Language), C, C++ and Java: There is a likeness between HTTP and SIP that allows to use CGI also to create services in the SIP environment. SIP CGI script like HTTP CGI runs in the server and passes message parameters through environment variables to a separate process. The process sends instructions back to the server through its standard output file descriptor.

3.3.2 SIP-Servlets

In contrast to a Java Applet which runs in a web browser a HTTP Servlet is a Java application that runs in a web server or application server and provides server-side processing, carrying out similar tasks like CGI(database access, e-commerce). But Servlets have another way to solve the tasks. Instead of using a separate process like CGI, Servlets pass messages to a class that runs within a JVM (Java Virtual Machine) inside the server, allowing all Java programming advantages. Consequently Servlets can support CGI or replace it.

A SIP Servlet does not interact with a HTTP Server but with a SIP server. It controls or influences call processing. The Server passes incoming SIP messages to the Servlet, which decides after controlling the message information(header, status) how to react to the message.

3.3.3 CPL - Call Processing Language

The Call Processing Language is a common language defining call processing logic. CPL works with both SIP or H.323 and can be used either on network servers or user agents. It is specified in RFC 2824 It is a programming language that fits completely into the XML structure and is used mostly to forward calls to a specific address or ignore them depending on the caller's address, the calling time or other circumstances.

Switches represent the choices, a cpl script can make - based on either attributes of the original call, or items independent of the call e.g. the time.

There are several Switches :

- address switches
- time switches

- priority switches

When a call establishment request arrives at a signalling server which is a CPL server, that server associates the source and destination addresses specified in the request with its database of CPL scripts; if one matches, the corresponding script is executed. So it is for example possible to ignore calls of specific persons or ignore all calls during the morning break - to receive a call on the mobile phone, per SMS or as a wave file per email - the decision is made by a CPL script.

3.4 Infrastructural Elements

3.4.1 University of Columbia CINEMA - The Server Component

As the core component of the SIP infrastructure, we have chosen the University of Columbia's Columbia InterNet Extensible Multimedia Architecture (CINEMA). It consists of a Proxy Server, a Conference Server, a Unified Messaging Server, a SIP-H.323 translator and a RTSP Media Server. The SIP Server is a standard conform Proxy offering Registrar and authentication services. We licensed the sipd Proxy Server. The SIP-H.323 translator sip323 can be used for free without license as a single call version.

The system is available for Windows 2000, NT 4.0, Sun Solaris 5.8, Linux (RedHat 6.2, RedHat 7.1 or later recommended), FreeBSD 4.3.

We have chosen their Server because it was the only one affordable for us. The perpetual license costs $100 for academic use. It SHOULD offer all the functionality required. In fact we had very many problems with it impeding the desired mastering of its configuration which is pretty complicated due to erroneous configuration scripts and a virtually non-existent administration manual.

Modularity

The modularity of the SIP system favors later amplification (as presented later in the section 3.4.1). The paragraph "Adding new services" concentrates on the amplification of the back-end infrastructure connecting the SIP network to other protocols or other networks like PSTN, while in the paragraph "Quasi-automatic incorporation of new ways of communication" the implementation of new technologies in the field of the front-ends (the user agents) is detailed.

Unfortunately, due to problems with our SIP server, we can only provide very basic functionalities. In addition to that most SIP phones only incorporate the basic call features. Due to the degree of novelty we encountered virtually no really mature product. Many of the programs we used had more than two releases during the time of the project, like the Windows Messenger for example.

Adding new services The basic infrastructure requires one server. This server usually incorporates the Proxy/Redirect services and the Registrar service. All extensions to the system connecting it to other services or protocols act like separate SIP user agents. The classic extensions are SIP/PSTN Gateways or SIP/H.323 Gateways. But voicemail servers can be implemented easily as well. See the section 4.1 on page 23 for more detailed description of the concrete possibilities to realize this.

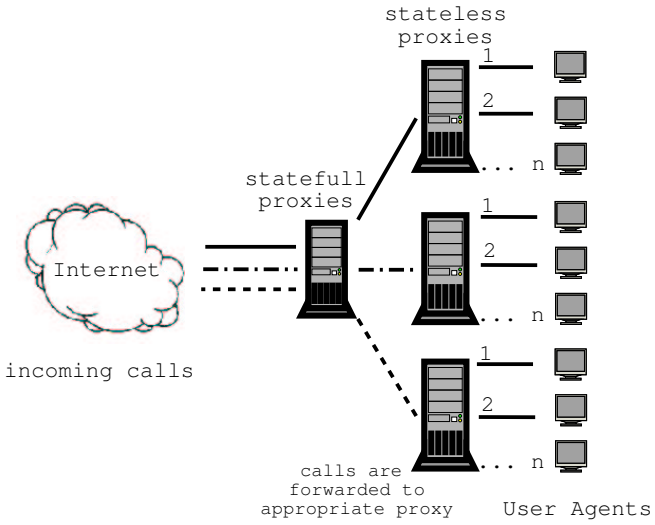
If the provider offers these services the users can define forwarding rules like: "If the telephone has rung for 30 seconds, forward the call (the INVITE method) to the following address). This address could be the address of a voicemail server which accepts the INVITE (answers with a 200 OK), plays a wave and records the incoming media. Afterwards, the user can connect to the voicemail server getting his messages. Each defined gateway accepts the INVITE and handles it in the correct manner like translating the SIP message to H.323 messages in the case of a SIP/H.323 Gateway or like extracting to phone number and calling the number in the case of a SIP/PSTN Gateway.

Quasi-automatic incorporation of new ways of communication Due to the fact that the inner SIP infrastructure (the servers) are only forwarding the messages, the changes in the underlying payload, introducing other, new services, are virtually irrelevant. For these, only the header fields are important (with exception to the REGISTER method where a CPL script can be sent to the server in the payload). The whole communication after session establishment is managed by the user agents based on the SDP information in the payload of the SIP message. So it's only a question if both user agent are able to realize the desired session type. If both user agents support video streaming with the same codec, the communication will be established. If both user agents support the launch of a Quake 3 session, even this session can be established. Actually, most of the SIP user agents "only" implement audio communication. So if only voice communication is available at the end of our project, this will not be a limitation in the realizable functionality spectrum of the infrastructure. When video communication, the usage of whiteboards, collaborative usage of the Desktop or other ways of communication become available, they can be integrated in the infrastructure simply be replacing the old user agents by the new ones. The website www.sipcenter.com present an article saying that Netmeeting will incorporate SIP. So its functionality will be available soon. This was the main reason for integrating Netmeeting into the website, although some problems with the current H.323/SIP gateway.

Scalability

Basic Performance The basic performance of the CINEMA sipd SIP Proxy Server is given through two performance indicators. Possible treatment of REGISTER methods and possible treatment of calls (INVITE method). According to the Webpage of the University of Columbia the sipd Server can handle a basic throughput of 100 REGISTER methods or the signalling of 20 calls per second, measured on a SUN workstation (the exact type wasn't given).

Figure 3.1: Scaling a SIP infrastructure via Proxy Chaining



Proxy Chaining One way to scale a SIP infrastructure is to use a technique called Proxy chaining as shown in figure 3.1. The basic SIP structure has only one Proxy. In this case the Proxy should be a stateful Proxy to offer enhanced call management like automatic repetition of messages or automatic switching to TCP if UDP doesn't work. Additionally this server has to run the CPL scripts or SIP Servlets to provide call processing logic. This decreases the server's ability to process requests simultaneously. The usage of more servers would have the disadvantage

of different SIP URIs for the user. In this case Proxy chaining is the appropriate solution. One outer server runs in stateless mode offering a higher request processing rate. The server has a complete database of the users within the domain with "hard-coded" (that means not modifiable by REGISTER) links to the Proxy servers where they register. So it simply forwards the incoming calls to the server where the respective user has his account. These servers offer the services mentioned above. So traffic is split into smaller parts. This method follows the rule to put the expensive logic to the rim of the network and to have the fast, light-weight forwarding units in the core of it. Due to the problematic configuration of the server, we could not evaluate how to set a Proxy Chain with sipd servers. But as the system is already quite performant there is no basic need to do this. You might stochastically evaluate the number of users which can be held on one single server. Therefore you will need to gather statistical data about the practical usage of the system which was impossible for us to achieve.

Billing

As a service is offered, billing issues may arise. Right now the tutorials are paid by the University, but maybe the added value through callcenter functionality should be paid for. So billing of calls should be provided. Basically this feature is already implemented in the server. It is possible to log all the messages received from the server. So you can keep track of the tutor, if they are available to the times contracted, as well as a trace of the users registering and making calls to the tutor. A billing function which rates the calls made can be defined. The tariffs can be defined by access level, time and caller. Automatic creating of invoices in not supported. The only currency supported is USD but this should not be a problem, as the invoices have to be created without the implemented billing function.

Billing SIP is a quite difficult issue. We experienced some problems with it during the tests. The basic logic behind the billing is to find the initiating INVITE and the corresponding BYE to determine the duration of a call. During the tests, due to errors and non RFC compliant behavior of some User Agents (UA) the calls were initiated but not terminated correctly. So the server cannot calculate the duration and fails to create a proper bill. There is a special rating page in the server configuration website, where the server show unrateable calls. So a method to make "free" calls would be to terminate calls by shutting down the client improperly. If the client of the callee does automatically finish the call on the collapse of the communication there won't be a proper billing. Some UAs like the eStara softphone try to complete the transaction in the background finishing the call properly. But on the other hand if the UA of the callee does not switch from call state to available state (that means the call remains present for the callee) if the communication breaks for external reasons, the caller maybe betrayed by the callee. He just has to wait and terminate the call minutes or even hours after the call. So the user would have to pay for minutes he was not on the line. This maybe detected by the server if it tracks the ACK messages of the caller but this opens another field for attacks.

So the billing is not very secure. If the call flow is kept, no problems arise. Small deviations might be detected but only making some assumptions which opens the field for unfairness.

3.4.2 User Agents

Microsoft Messenger

The Microsoft Messenger included in Windows XP is using the SIP protocol. It is offering a huge set of possibilities. Besides the instant messaging and online awareness part like ICQ, you can establish phone calls, video calls, file transfers, a collaborative whiteboard and the initiation of game session via the internet.

In its actual version 4.6 it interoperates quite well with the SIP Server. The only constraint is that its usage is limited to the users of Windows XP.

Pingtel instant xPressa

Instant xPressa is not just a regular SIP webphone but an emulation of an existing hardware SIP phone, the Pingtel xPressa. It was the world's first JAVA VoIP telephone. It is programmable via a development kit offered by Pingtel. These small applications are xPressions. They can be uploaded to the hardware phone as well as integrated in the softphone.

The telephone is configured via a build-in web server.

It was one of the two native SIP phones, but unfortunately the phone has problems to authenticate with the server. It can manage the used Digest Access Authentication Scheme, but it responds only to the 407 Proxy Authentication Required response from the server. It cannot handle the 401 Unauthorized Access response sent by sipd.

Ubiquity

The Ubiquity Helmsman SIP User Agent supports redirection, authentication and up to 5 simultaneous calls. Helmsman works only with Windows 95/98/NT(/2000/XP tested by us and they work as well). Although it is implemented in JAVA its usage is limited to the Windows systems.

eStara Softphone

The eStara Softphone claims to be the only market-ready web voice service truly designed and ready to meet the needs of doing business online. They establish their claim by emphasizing the fast and easy installation because of its firewall accommodation and the easy integration in call centers. It supports collaborative web browsing in the commercial edition. The free evaluation version has problems with the Digest Authentication. It offers the best integration for Windows, because it reconfigures the Internet Explorer to forward sip: links to the phone.

If the security standard is lowered, the phone can be used.

HistoryClient

This little tool was developed by us to have an equivalent to the functionality of traditional callcenter systems regarding their ability to display information in a automatic way. The HistoryClient User Agent will do this by popping up a window with a history of all calls documented with the caller as soon as the INVITE message comes in. So the tutor has an instant overview of the problems of this caller.

To fulfill this, the HistoryClient logs on to the server automatically after starting it using the same account data as the tutor but registering to a different port. When the tutor is called the server forks the call to both User Agents registered to the tutor (the phone and the HistoryClient). Then it opens the history file and displays the notes in chronological order. The tutor can add a documentation for the current call or edit earlier call descriptions.

The client is a small tool following the KISS design principle. It offers a small but nevertheless useful functionality. Also it can be viewed as a feasibility study for further implementations, because the implementation of the authentication procedure is quite tricky and cost us much effort and time (see section 4.1.2). It can be easily reused or at least taken as an example for the basic communication with the server including the authentication methods.

Chapter 4

Evaluation of The Implemented Infrastructure

The implemented infrastructure as presented in figure 4.1 on page 24 is a classic SIP infrastructure which provides the best interoperability with other SIP systems.

4.1 Interoperability

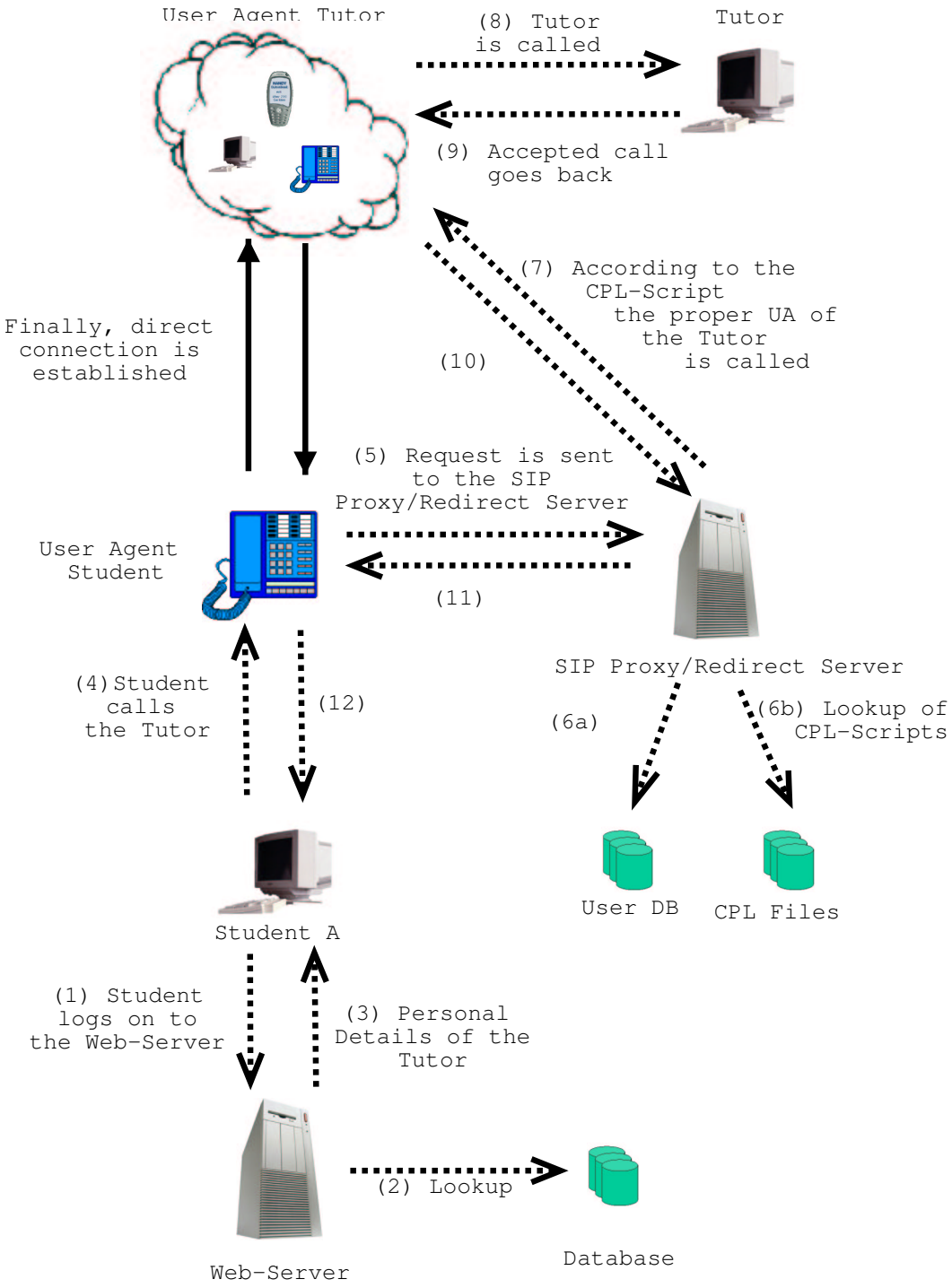
4.1.1 PSTN connectivity

The cloud in figure 4.1 shows ALL possibilities of the SIP system. Actually we will have to stick to the possibilities of a software user agent. Interoperability with PSTN-Phones or GSM Phones is only possible via SIP/PSTN-Gateways. These Gateways are hardware consisting of a DA/AD converter for the sound, a network card for internet connection, usually an ISDN adapter for the connection to the PSTN network and the SIP interpreter. So the calls are forwarded to the gateway, the interpreter determines the number to be called, calls the number and handles the media streams of the following call. Due to the fact that it is hardware the investment is quite expensive in relation to the benefit. Another solution would be to contract a service provider. There are very few providers offering these services. One constraint is that they are commonly using H.323. This could be solved using H.323/SIP-Gateways. The decisive constraint is the fact that all these providers are using their own softphones. The whole call architecture is hidden to the user. We tried to contact the providers to get the details in order to configure our infrastructure adequately but only very few answered. In most of the cases they are prohibiting the usage of their services. Others are using a kind of "hacked" version of the H.323 protocol family. Surprisingly only one provider offers a regular SIP/PSTN-Gateway service, but only for his proprietary softphone which is not interoperable with standard SIP Servers.

Interoperability to Vovida Vocal and H.323 Vocal is a SIP stack developed by Vovida which was bought by Cisco Systems and made Open Source by them. Vocal implements a special infrastructure. For a detailed description read the Milestone 1 regarding this point. The main difference is, that you only have the Redirect functionality in the server. To offer proxylike behavior of the system, the user agent doesn't communicate directly with the server but via a new component, the Marshal Server which forwards the messages, intercepting and evaluating the redirect messages of the Redirect Server.

With a regular SIP Proxy, interoperability with Vocal can accomplished. So one wish criterion was the connection of our infrastructure with the Vocal system to use services like the included voicemail server. See section 3.4.1 on page 18 for a basic description of the possible integration process. Due to the delay of the installation of the SIP server we had to cancel the integration.

Figure 4.1: Proposed Infrastructure with a call flow example



4.1.2 Conclusion on The Interoperability

The constraints mentioned above are extremely limiting the possible communication ways of our infrastructure. But it doesn't limit the SIP functionality itself due to the fact that SIP is clearly separated from the underlying communication. So the infrastructure is easily extensible in the case that later someone wants to spend the money for a hardware PSTN-Gateway or more probable a provider opens his services to regular SIP user agents. In that case only a forwarding rule has to be adjusted and the Gateway is integrated.

But interoperability is often only a virtual concept. In fact, it is very fragile. We started with 8 User Agents to test. But the problem for most of them was the not implemented Digest Access Authentication Scheme. So the only UAs which manage this, are the Ubiquity Helmsman, the Windows XP Messenger and the Pingtel xPressa. The last phone name left the circle of possibilities by requiring the 407 response (see section 3.4.2 on page 21 for further details). When we implemented the Digest Access Authentication by ourselves for the HistoryClient we had pretty much problems to implement a working version. Our research led to the conclusion that it is rarely used in the HTTP world for which it originally was designed, so there are hardly no resources in the net to find out how it works. An alternative is surely the lowering of the security standards. Then the circle of interoperable phones is significantly increased. Especially the eStara Softphone is a valuable phone to use due to its good integration in Windows.

The Windows Messenger has actually some problems to establish a call to other SIP phones, because it uses different audio parameters which the other UAs cannot provide (namely the GSM standard and G723 in opposition to the PCMU 8kHz the other UAs use). The parameters can't be modified manually so the signalling works, but the other side cancels the call due to incompatibility.

Fully interoperable are the Pingtel xPressa and the Ubiquity Helmsman.

While implementing the HistoryClient we realized also that the server is very intolerant regarding the composition of messages. They have to be extremely exact according to the specification. If not formulated well they are simply denied, although the 400 Bad Request message returned shows that the server was able to parse the request correctly and extracted the correct information out of it.

4.2 Final Words

SIP will be the future of real-time communication via the Internet. Proprietary standards will be replaced by SIP. But the problems we encountered especially during the implementation of the SIP part of the infrastructure show that the whole construct is still in the first evolution phase. But the possibilities will increase. Best example is the entering of Microsoft into the SIP market with the Messenger. With the proliferation of Windows XP the number of SIP users will increase singnificantly. This demand will force new publicly accessible SIP accounts. Your SIP address will become as important as your eMail address is right now.

Appendix A

MySQL

The following list describes some of the important characteristics of MySQL:

- Fully multi-threaded using kernel threads. That means it easily can use multiple CPUs if available.
- C, C++, Eiffel, Java, Perl, PHP, Python and TCL APIs. 20 MySQL client tools and APIs.
- Works on many different platforms. Many operating systems supported by MySQL.
- signed/unsigned integers 1, 2, 3, 4 and 8 bytes long, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET and ENUM types. 7.2 Column types.
- Very fast joins using an optimized one-sweep multi-join.
- Full operator and function support in the SELECT and WHERE parts of queries. Example: mysql62; SELECT CONCAT(first_name, " ", last_name) FROM tbl_name WHERE income/dependents 62; 10000 AND age 62; 30;
- SQL functions are implemented through a highly-optimized class library and should be as fast as they can get! Usually there shouldn't be any memory allocation at all after query initialization.
- Full support for SQL GROUP BY and ORDER BY clauses. Support for group functions (COUNT(), COUNT(DISTINCT), AVG(), STD(), SUM(), MAX() and MIN()).
- Support for LEFT OUTER JOIN with ANSI SQL and ODBC syntax.
- You can mix tables from different databases in the same query (as of version 3.22).
- A privilege and password system which is very flexible and secure, and which allows host-based verification. Passwords are secure since all password traffic when connecting to a server is encrypted.
- ODBC (Open-DataBase-Connectivity) for Windows95 (with source). All ODBC 2.5 functions and many others. You can, for example, use Access to connect to your MySQL server. 16 MySQL ODBC Support.
- Very fast B-tree disk tables with index compression.
- 16 indexes per table are allowed. Each index may consist of 1 to 16 columns or parts of columns. The maximum index length is 256 bytes (this may be changed when compiling MySQL). An index may use a prefix of a CHAR or VARCHAR field.

- Fixed-length and variable-length records.
- In-memory hash tables which are used as temporary tables.
- Handles large databases. We are using MySQL with some databases that contain 50,000,000 records.
- All columns have default values. You can use INSERT to insert a subset of a table's columns; those columns that are not explicitly given values are set to their default values.
- Uses GNU Automake, Autoconf, and libtool for portability.
- Written in C and C++. Tested with a broad range of different compilers.
- A very fast thread-based memory allocation system.
- No memory leaks. Tested with a commercial memory leakage detector (purify).
- Includes isamchk, a very fast utility for table checking, optimization and repair.
- Full support for the ISO-8859-1 Latin1 character set.
- All data are saved in ISO-8859-1 Latin1 format. All comparisons for normal string columns are case insensitive.
- Sorting is done according to the ISO-8859-1 Latin1 character set (the Swedish way at the moment). It is possible to change this in the source by adding new sort order arrays. To see an example of very advanced sorting, look at the Czech sorting code. MySQL supports many different character sets that can be specified at compile time.
- Aliases on tables and columns as in the SQL92 standard.
- DELETE, INSERT, REPLACE, and UPDATE return how many rows were changed (affected).
- Function names do not clash with table or column names. For example, ABS is a valid column name. The only restriction is that for a function call, no spaces are allowed between the function name and the '(' that follows it.
- All MySQL programs can be invoked with the –help or -? options to obtain online assistance.
- The server can provide error messages to clients in many languages.
- Clients connect to the MySQL server using TCP/IP connections or Unix sockets, or named pipes under NT.
- The MySQL-specific SHOW command can be used to retrieve information about databases, tables and indexes.

Appendix B

myPhpAdmin Screenshots

Figure B.1: The starting interface of phpMyAdmin

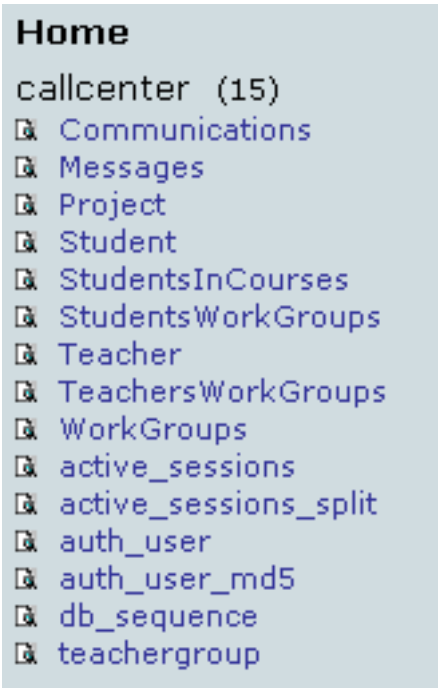
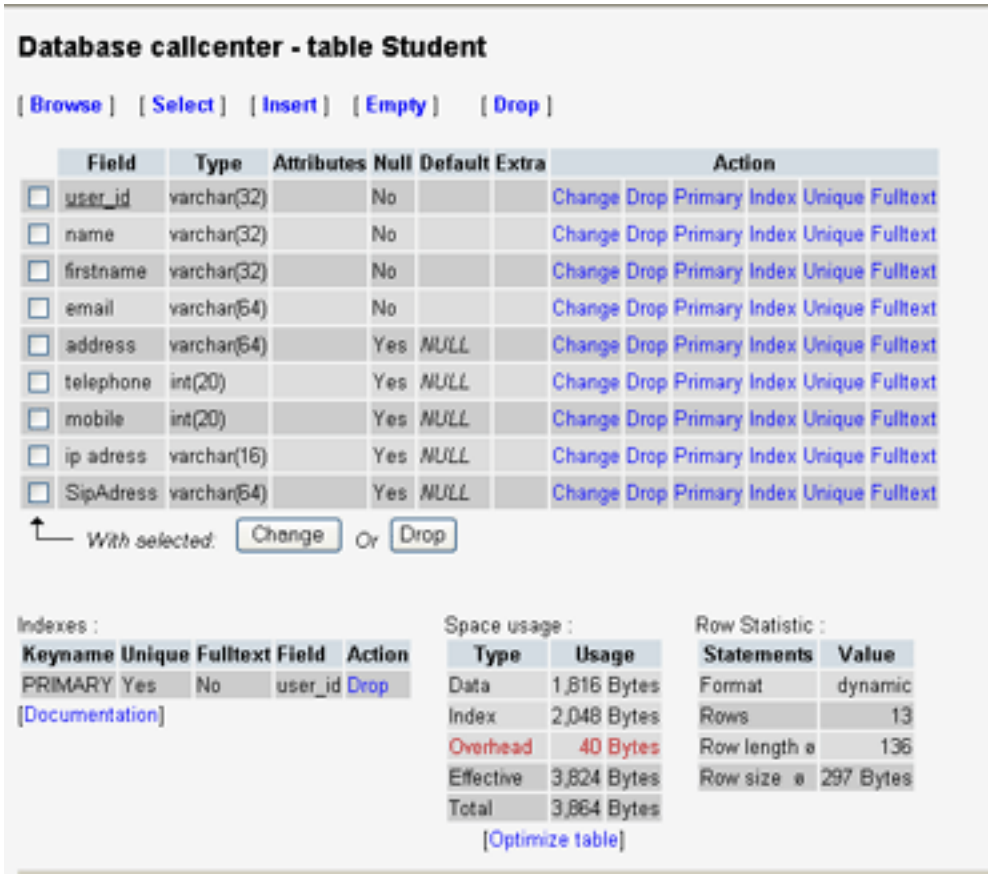


Figure B.2: On the right of the screen there are the different tables of the database



Database callcenter - table Student

[Browse] [Select] [Insert] [Empty] [Drop]

	Field	Type	Attributes	Null	Default	Extra	Action
☐	user_id	varchar(32)		No			Change Drop Primary Index Unique Fulltext
☐	name	varchar(32)		No			Change Drop Primary Index Unique Fulltext
☐	firstname	varchar(32)		No			Change Drop Primary Index Unique Fulltext
☐	email	varchar(64)		No			Change Drop Primary Index Unique Fulltext
☐	address	varchar(64)		Yes	NULL		Change Drop Primary Index Unique Fulltext
☐	telephone	int(20)		Yes	NULL		Change Drop Primary Index Unique Fulltext
☐	mobile	int(20)		Yes	NULL		Change Drop Primary Index Unique Fulltext
☐	ip adress	varchar(16)		Yes	NULL		Change Drop Primary Index Unique Fulltext
☐	SipAdress	varchar(64)		Yes	NULL		Change Drop Primary Index Unique Fulltext

With selected: Change Or Drop

Indexes :

Keyname	Unique	Fulltext	Field	Action
PRIMARY	Yes	No	user_id	Drop

[Documentation]

Space usage :

Type	Usage
Data	1,816 Bytes
Index	2,048 Bytes
Overhead	40 Bytes
Effective	3,824 Bytes
Total	3,864 Bytes

[Optimize table]

Row Statistic :

Statements	Value
Format	dynamic
Rows	13
Row length ø	136
Row size ø	297 Bytes

Figure B.3: Display of the table schemes

Appendix C

Implementation of PHPLib and Class Diagram

Callcenter_Session

```
class Callcenter_Session extends Session {
  var $classname = "Callcenter_Session";

  var $cookiename     = "";                ## defaults to classname
  var $magic          = "Hocuspocus";      ## ID seed
  var $mode           = "cookie";          ## track session IDs with cookies
  var $fallback_mode  = "get";
  var $lifetime       = 0;                 ## 0 = cookies, else minutes
  var $that_class     = "Callcenter_CT_Sql"; ## name of data storage container
  var $gc_probability = 5;
}
```

Callcenter_CT_Sql

```
class Callcenter_CT_Sql extends CT_Sql {
  var $database_class = "DB_Callcenter";
  var $database_table = "active_sessions";
}
```

`$database_class` defines the database class wich is used by Session

`$database_table` name of table where the Session data is stored

DB_Callcenter

```
class DB_Callcenter extends DB_Sql {
  var $Host     = "localhost";
  var $Database = "callcenter";
  var $User     = "callcenter";
  var $Password = "elearning";
}
```

`$Host` the Host of the Callcenter Database

`$Database` the Name of the Callcenter project database

`$User` the username for the connection to the database server

`$Password` the password for the connection to the database server

Callcenter_Auth

```
class Callcenter_Auth extends Auth {
  var $classname      = "Callcenter_Auth";

  var $lifetime       =  15;

  var $database_class = "DB_Callcenter";
  var $database_table = "auth_user";

  function auth_loginform() {
    global $sess;
    global $_PHPLIB;

    include($_PHPLIB["libdir"] . "callcenter_loginform.ihtml");
  }

  function auth_validatelogin() {
    global $username, $password;

    if(isset($username)) {
      $this->auth["uname"]=$username;    ## provide access to "loginform.ihtml"
    }


    $uid = false;

    $this->db->query(sprintf("select user_id, perms ".
                             "        from %s ".
                             "       where username = '%s' ".
                             "         and password = '%s'",
                        $this->database_table,
                        addslashes($username),
                        addslashes($password)));

    while($this->db->next_record()) {
      $uid = $this->db->f("user_id");
      $this->auth["perm"] = $this->db->f("perms");
    }
    return $uid;
  }
}
```

`$classname` this contains the classes name

`$lifetime` defines the duration of authentification if the after this time the user is loged out automatically

`$database_class` defines the database class wich is used by Session

`$database_table` name of the table where the authentification information will be stored

`function auth_loginform()` used to display the loginscreen defines the .ihtml file which contains the HTML form for the login

`function auth_validatelogin()` this function validates the login information

Callcenter_Perm

```
class Callcenter_Perm extends Perm {
  var $classname = "Callcenter_Perm";

  var $permissions = array(
                            "user"       => 1,
                            "author"     => 2,
                            "editor"     => 4,
                            "supervisor" => 8,
                            "admin"      => 16
                          );

  function perm_invalid($does_have, $must_have) {
    global $perm, $auth, $sess;
    global $_PHPLIB;

    include($_PHPLIB["libdir"] . "callcenter_perminvalid.ihtml");
  }
}
```

`$classname` This contains the classname

`$permissions` this array contains the available user permissions, as every user can have multiple user rights these values in the array are interpreted as bits. The user rights of each user is stored in the auth_user table

`function perm_invalid($does_have, $must_have)` this function is used to display the error message in callcenter_perminvalid.ihtml if an user has insufficient rights to access a page

Callcenter_User

```
class Callcenter_User extends User {
  var $classname = "Callcenter_User";

  var $magic          = "Abracadabra";     ## ID seed
  var $that_class     = "Callcenter_CT_Sql"; ## data storage container
}
```

`$classname` refers the classname

`$magic` this variabele contains the ID seed

`$that_class` defines the data storage container for the user class

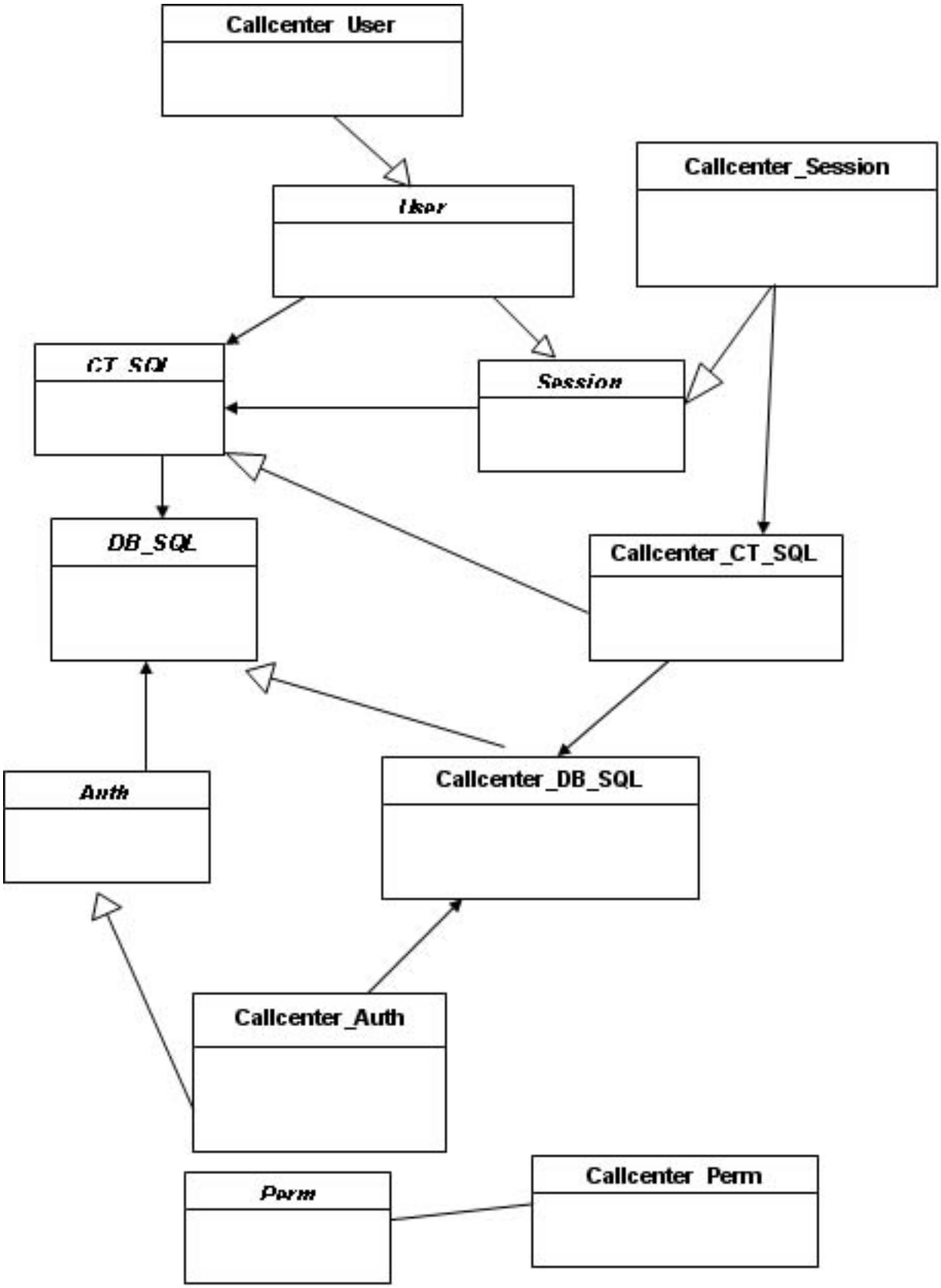


Figure C.1: Class Diagram for the PHPLib files of the website