



POLITECHNIKA ŚLĄSKA W GLIWICACH

Tytuł pracy: System obsługi Państwowego Szpitala dla Psychicznie i Nerwowo Chorych w Rybniku

Student: Zbigniew JERZAK

Promotor: Dr Jarosław FRANCIK

Nazwa uczelni: Politechnika Śląska w Gliwicach

Instytut: Informatyki

Gliwice, sobota, 11 października 2003



System Dystrybucji Dokumentów

użytkownik:

hasło:

web doc

wersja 1.50

TABLE OF CONTENTS

Foreword.....	5
Contents	6
1. CHAPTER ONE – PATH TO WEBDOC	7
1.1 Summary	8
1.2 Short Story about Paper	8
1.3 Paper as Information Interchange Media	10
1.4 The Need for EDI.....	11
1.5 What is EDI?.....	11
1.6 How does EDI look like?.....	12
1.7 EDI Architecture.....	13
1.8 Costs of EDI.....	14
1.9 A Word about XML	14
1.10 XML and Business Applications	15
1.11 Combining XML and EDI	16
1.12 WEBDOC – an Alternative	17
1.13 What is WEBDOC exactly?	18
1.14 WEBDOC and Hospital Information Systems	18
1.14.1 Hospital Information Systems.....	19
1.14.2 WEBDOC in HIS environment.....	21
2. CHAPTER TWO – THE DATABASE.....	22
2.1 Summary	23
2.2 SAPDB.....	23
2.3 PostgreSQL.....	23
2.4 MySQL.....	24
2.5 Why InnoDB?.....	25
2.6 Decision	26
2.7 Database description.....	28

2.7.1 Database description - users table	28
2.7.2 Database description - files table.....	28
2.7.3 Database description - groups table	29
2.7.4 Database description - ACL table	30
2.7.5 Database description - GACL table	31
2.7.6 Database description - uglobals table.....	31
2.7.7 Database description - userhistory table	32
2.7.8 Database description - filehistory table.....	32
2.7.9 Database description - keywords table.....	33
2.7.10 Database description - emailnot table	33
2.8 Database specific data	34
2.9 Database diagram	35
3. CHAPTER THREE – THE SYSTEM	36
3.1 System Highlights	37
3.2 PHP and HTML Pages	42
3.3 How to navigate files.....	44
3.4 How to navigate code	47
4. CHAPTER FOUR – THE SERVER	50
4.1 Summary	51
4.2 Server security.....	51
4.3 Installation guide.....	54
4.3.1 Installation guide - Server, Apache and PHP	54
4.3.2 Installation guide - MySQL Database	55
5. CHAPTER FIVE – CONCLUSIONS AND THE FUTURE.....	56
5.1 Possible future development	57
5.2 Conclusion	58
6. REFERENCES	59
7. ATTACHEMENTS	62

Foreword

The aim of this Master Thesis was developing a system for storage, distribution and management of documents in a hospital environment. One of many factors that resulted in developing the system were prospected reduction of paperwork, improvement in document circulation and enhanced accessibility of documents, regardless of the system's client location. Herewith WEBDOC system introduces a secure user and user's rights management database. Thanks to the well thought implementation of security policies this systems proves to be applicable in every environment were data security is of high importance. Thanks to its open architecture it can be easily integrated into other systems, thus allowing to increase security of document circulation.

It has to be stressed that WEBDOC is not a Hospital Information System (HIS) in a strict sense of this word. It provides a means of distribution and securing electronic documents circulating in the hospital's LAN. WEBDOC does not implement Management Information System nor Financial Information System functionality. However, thanks to it's generic approach it proves to be very useful in this area. It complements the above mentioned systems in very effective and effortless way.

The resulting system has been successfully implemented at the State Psychiatric Hospital in Rybnik, Poland and at the Fachhochschule Aalen, Aalen, Baden-Wuerttemberg, Germany. Moreover it is planned to implement the system at the FH Ulm, Ulm, Baden-Wuerttemberg, Germany. This work has been originally inspired by professor Manfred Strahnen from the Fachhochschule Aalen. It would not be possible to accomplish this project if it has not been for the valuable help and support from Ph.D. Jarosław Francik from the Silesian University of Technology in Poland. This system's idea stems also from the work of Marcus Fetzer, a student of FH Aalen.

Herewith I would also like to thank the director of the State Psychiatric Hospital in Rybnik – Mr. Stanisław Urban for his support and enthusiasm for this project.

The system is designed to run on both Linux and Windows Server platforms; however the thorough testing and debugging was conducted in the Linux environment – mainly due to the GPL and security issues. The system bases on the LAMP (Linux, Apache, MySQL, PHP) Web server system, as originally requested by the customer.

Current WEBDOC system development was focused on needs and requests presented by the State Psychiatric Hospital in Rybnik. However, system's flexibility and expandability allowed it to be successfully implemented at the FH Aalen. WEBDOC architecture and functionality allows it for a very effective integration into already existing or just emerging structures of any Hospital Information System.

Contents

This document focuses on explaining several important issues connected with the WEBDOC document distribution system. In the first chapter it is explained how the need for such an approach has been formed throughout the years of development of computer systems. Starting with the increasing paper consumption and going through the early concepts of the Electronic Document Interchange the first chapter introduces WEBDOC as an alternative to the already existing systems, highlighting its main differences in comparison to the already functioning solutions. In the first chapter one can also find the path an idea of paperless office has been evolving along, including such milestones as EDIFACT and XML.

The second chapter deals with the issues of the database selection and its implementation and design. At the beginning reader is presented with the spectrum of existing databases meeting the pre-selection criteria. Each of them is briefly described in order to finally highlight the one that has been chosen as the one being most applicable in the WEBDOC environment. This decision is than thoroughly documented and explained. Later in the chapter reader can get an insight in the database design with the detailed description of each table used in the system. Moreover, one will find some background information on how and why such structure has been formed. Finally this chapter addresses some database specific issues that are characteristic to the database used in the system. Last, but not least, reader is presented with the complete database diagram.

In the third chapter we can get a full insight into the system functioning and design. At the beginning reader can acquaintance himself with the Graphical User Interface of the system, along with the system's functionality. Later, an overview of how the system is implemented at the PHP and HTML level is presented. Code snippets provide useful information on how to navigate the code and how to be able to extend the system to meet the specific customer requirements – as it has taken place in case of State Psychiatric Hospital In Rybnik.

Fourth part is focused on the server implementation and deals with all the technical details on how to implement the system on the specific platform and how to ensure its stability and flawless functioning. This chapter deals in detail with security issues that are of great importance due to the specific system's working environment. At the end of the chapter reader can find a step-by-step installation guide aimed at the LAMP target platform.

Fifth part includes the ideas and concepts that were not included in the current release, however are likely to be introduced in future versions. Apart from that author has provided general conclusions on how did the system fulfilled the expectations set upon it. Once again an overview of system's flexibility allowing it to be successfully implemented in two very different, yet very demanding environments, is presented.

At the end of this work author has included a list of references along with all the attachments. Among the attachments are: the complete source of the WEBDOC project, the opinion presented by the CEO of the State Psychiatric Hospital in Rybnik and a short summary in Polish.

1. CHAPTER ONE – PATH TO WEBDOC

1.1 Summary

In this chapter author will try to indicate and emphasise circumstances that led to invention of the WEBDOC system idea. It is striking how much paperwork and paper based information is being exchanged in the contemporary world. There have been numerous attempts dating back as early as the 70s to reduce the usage of paper by introducing the concept of Electronic Document Interchange. EDI has successfully settled itself in a few branches of industry, however due to high costs of implementation it has not been as widespread as it has been awaited. Hence there are numerous attempts to enhance the EDI functionality along with reducing the costs of its usage. In this chapter it will be described how WEBODC fits in the void left by EDI as far as SBUs are concerned.

1.2 Short Story about Paper¹

Throughout the centuries human civilization development based on the invention of paper and on the use of paper as a method of transmitting and exchanging information. Starting with the invention of font made by Johannes Gensfleisch zum Gutenberg European civilization started its inevitable growth and expanse. Use of paper and paper based information became more extensive along with the increasing number of enterprises and different companies.

Global paper consumption has increased by a factor of twenty this century, and by a factor of three in the last three decades alone. Average per capita consumption varies widely, from 333 kg/person in the US, to 160 kg/person in Western Europe and 12 kg/person on average in the developing world.

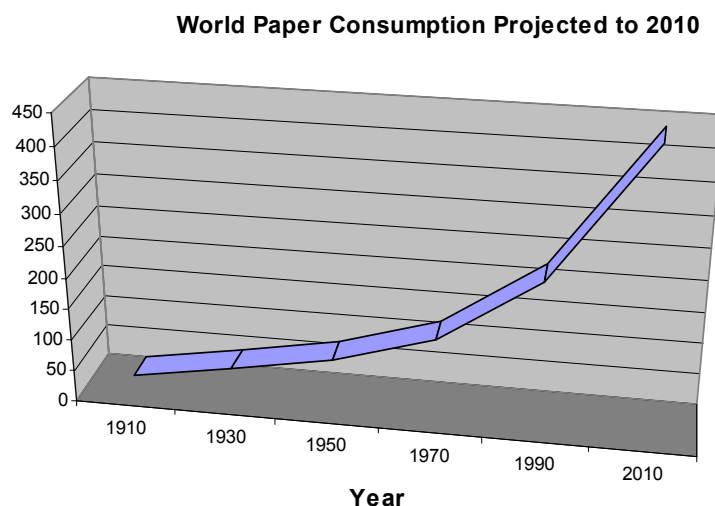


Fig. 1 - World paper consumption²

¹ The following chapter's data stem mainly from International Institute for Environment and Development Head Office - 3 Endsleigh Street, London WC1H 0DD, UK and from report of the OECD workshop - "Rethinking Paper Consumption", Oslo, Norway, 14 November 1996.

During this same period, growth rates in paper use among developing countries have been double that of the industrialised world. Although pulp and paper prices are extremely volatile, the long-term trend over the past thirty years has been for stable or declining real prices, suggesting that sufficient supply has been mobilised to meet growing demand. Today's annual average consumption of 48 kilograms per person conceals wide variations within and between countries. North America accounts for over one third of global consumption and Europe for over a quarter.

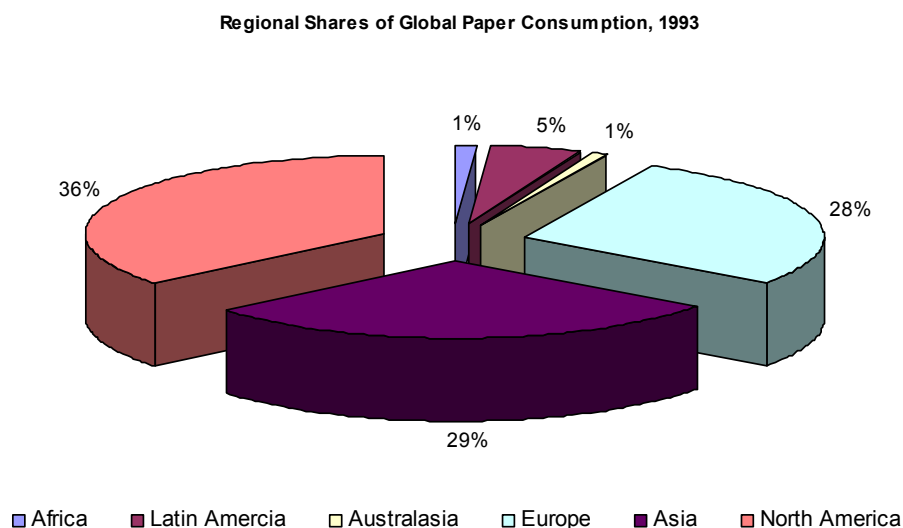


Fig. 2 - Regional Shares of Global Paper Consumption³

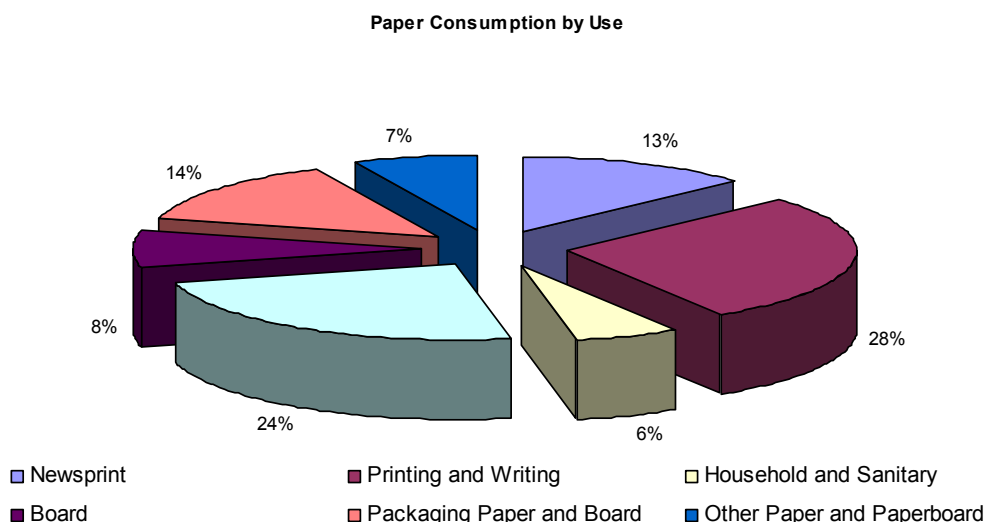


Fig. 3 - Paper Consumption by Use⁴

² According to Jaakko Pöyry 1999 paper market study

³ After www.economagic.com - PPI: Pulp, paper, and prod., ex. bldg. paper

Although paper is traditionally identified with reading and writing, communications has now been replaced by packaging as the single largest category of paper use. Only a small proportion is used for personal care products (see Paper Consumption by Use chart above). Exact proportions vary from country to country. For example the proportion of national paper consumption which is used for newsprint is around 20% in the UK, Canada and India, 26% in Tanzania, about 15% in the USA and Zimbabwe, and less than 5% in Uruguay and China.

1.3 Paper as Information Interchange Media

For centuries, paper was mainly used to store and communicate information. Today this end-use accounts for less than half of all paper consumed. However, this reflects increasing alternative uses for paper rather than an absolute decline in its use as an information carrier. The volume of paper used for printing and writing continues to grow at about 20% each year, and some estimates put the increase in the amount of paper used in offices at 600% over the past 25 years. Rather than leading to a reduction in paper use, the *information revolution* has been complementary, generating a surge in paper consumption, led by the diffusion of office computing, printing and copying technologies. Furthermore, the need to communicate more frequently and in greater detail has more than compensated so far for any displacement of paper by electronics. The paperless office, once predicted as a result of information technology (IT), has not transpired. Industry analysts estimate that 95% of business information is still stored on paper.

Increasing accessibility to copying machines, printers and fax machines fuel the demand for paper. There is a close correlation between paper sales and personal computers, copiers and desk top printing equipment sales, and the trend is up. The PC market continues to surge ahead, with a predicted growth of sales of 30% in Europe for 1995 alone; the number of photocopiers in Europe is expected to grow by 50% between 1993 and 1997. The next wave of printing technology is for colour printers, which again is likely to press paper consumption upward, while the introduction of plain paper faxes is likely to reduce the cost of fax paper, but not the quantity.

The increasing use of CD-ROMs does appear, however, to be substituting for paper use. Almost all desktop computers shipped in 1995 had built-in CD-ROM drives. A single CD can contain the equivalent of 330,000 sides of single-spaced text, and therefore constitute a *potential* threat to "hard copy". In 1995, sales of CD-ROM encyclopaedias in the USA outstripped printed editions for the first time, according to Microsoft. Encyclopaedia Britannica introduced a single CD-ROM holding the 44 million words of the 32 volume encyclopaedia in October 1994, and it has already captured around 25% of the UK market (and over 30% of the US market). Since the move to CD-ROM allows an escape from the space constraints of the printed book, updating is much easier on the electronic version. Indeed, the hard copy is so much harder to update that there is growing pressure to abandon it altogether.

⁴ After www.economagic.com - PPI: Pulp, paper, and prod., ex. bldg. paper

1.4 The Need for EDI

There were four main factors that spoke for the need of creating a system for electronic document interchange. First one of them is a fact that 70% of computer output is re-keyed into another computer. Second one is the reduction of paper use and protection of the natural environment – as stated during the OECD workshop - "Rethinking Paper Consumption" that took place in Oslo, Norway on the 14th of November 1996. Another one is the need to accelerate the document interchange and finally last, but not least document's security issues are also to be taken into account.

The problem of re-keying information into computers is very common when two different business units exchange information. Even if information is provided in the electronic form there usually exists no corresponding datasheet at the receiving part. The same occurs if the data exchange direction switches. This issue impacts mainly the speed of data processing and data interchange since usually re-typing requires thorough understanding of both sides forms.

Second problem needing to be stressed is the security issue. It is quite foreseeable that conventional paper based confidential information is intercepted on its way to recipient. With electronic media even the interception does not guarantee the attacker the ability to modify or read the information.

All these concerns were the foundation on which the concept of Electronic Document Interchange grew.

1.5 What is EDI?

EDI is an acronym that stands for "Electronic Data Interchange". While the term EDI refers to a general group of electronic business languages, there are many specific languages within the group that are notable, such as ANSI (American National Standards Institution) X12⁵, HL7 - Health Level 7 and EDIFACT⁶. EDI could be summarized as a term used to describe standards for proprietary electronic business messages.

EDI was first used in the transportation industry in the 1970's. Ocean, motor, air, rail carriers, and the associated shippers, brokers, customs, freight forwarders, and bankers put EDI into practice. The first set of EDI standards were developed by the Transportation Data Coordinating committee (TDCC). Since then, there has been a proliferation of EDI standards around the world.

EDI languages are usually written by a group of users with a common interest in developing a vocabulary for their particular vertical domain. HL7 is a good example of this. A group of stakeholders wanting to circulate messages within the health domain collaborated and agreed on a standard for a

⁵ On March 5, 2003 the Accredited Standards Committee (ASC) X12 approved a strategic direction that embraces collaboration with domestic and international organizations while continuing to forge ahead to ensure ASC X12 member companies' electronic data interchange (EDI) requirements are met. ASC X12 committed in its new strategy to continue to clarify its message design architecture, XML schema syntax and XML design rules and guidelines. ASC X12 also intends to harmonize X12 XML business messages with UN/CEFACT's approach.

⁶ Electronic data interchange for administration, commerce and transport (EDIFACT) is described in the ISO 9735 standard.

set of messages that they could use. Another example could be SWIFT an international money transfer system, that is used commonly in Europe.

1.6 How does EDI look like?

EDI is a highly structured set of messages and these messages are often very cryptic to an outside observer. The following message is an ANSI X12 4040 850 Purchase Order example:

```
ST*850*0001|
BEG*00*SA*A123456**19980507|
DTM*002*19980514|
N1*ST**9*1122334450000|
N2*ABC Co.*Branch 1|
N3*1234 Maple Grove St.|
N4*Apple Grove*CA*98765|
PO1*1*6*EA*5.74**
BP*00054321876547|
DTM*002*19980522|
PO1*2*12*EA*8.7**
BP*00054321647437|
PO1*3*12*EA*10.75**
BP*00054321674310|
SE*12*0001|
```

The structure of an EDI document is very simple. Basically, each line represents a "Data Segment". The letters at the beginning of each line are called the "Segment ID". A data segment is the intermediate unit of information in a message. A segment consists of a pre-defined set of functionally related "Data Elements" that are identified by their sequential positions within the set. Segment ID's are often unique 2 or 3 character alphabetic upper-case codes. EDIFACT uses upper case letters only while X12 uses a 2 or 3 character code composed of upper-case characters and digits. The Segment ID's are unique within each message and each segment ends with a *Segment Terminator*. A dedicated Segment terminator character is essential in the event that a long line gets wrapped. When data segments are combined to form a message, their relationship to the message is specified by a Data Segment Requirement Designator and a Data Segment Sequence. An EDI data segment is analogous to a logical record.

On the line after "BEG" in the example above, there are a series of values delimited by an asterisk (*). In X12, the asterisk is commonly used as the "Delimiter" meaning that it marks the boundaries between the values that are used by that segment. Elements are referenced by combining the Segment ID with a number that represents the count in a left to right direction. "BEG 01" would refer to the 1st piece of data, bound in between the first and second asterisks, in the BEG Segment. In that example, the value of BEG 01 is "00".

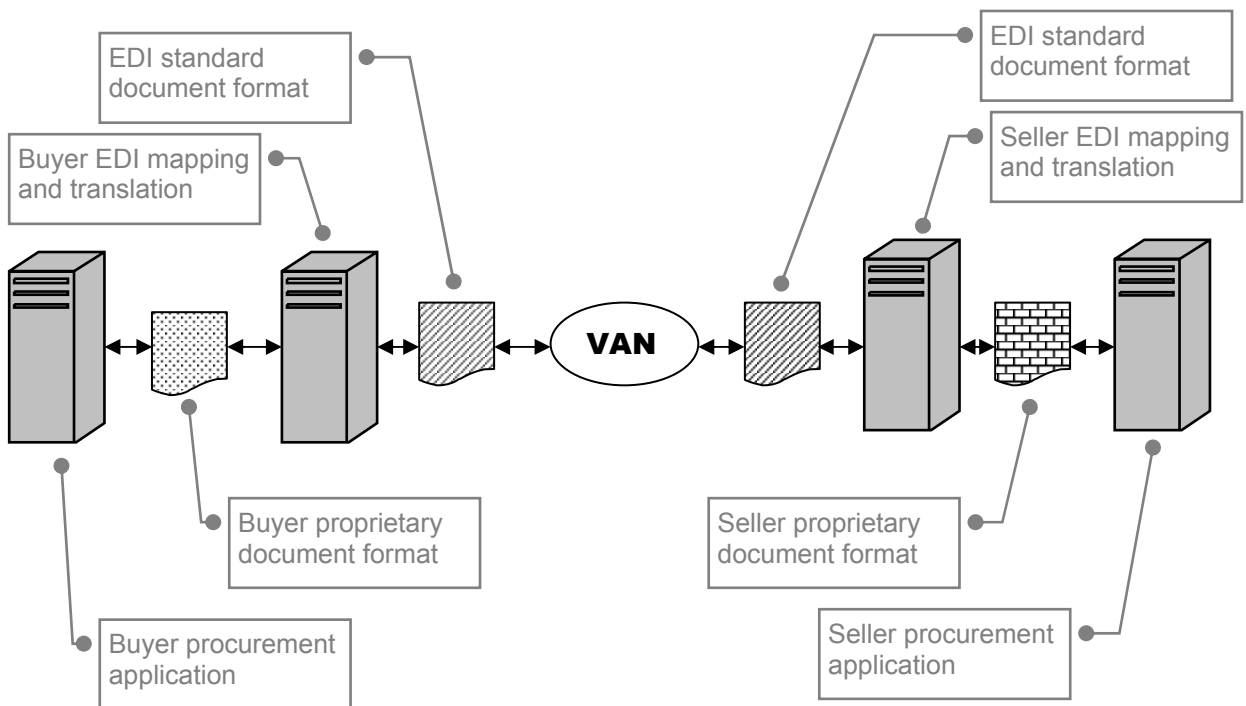
To determine the semantics of each value represented, one needs to refer to the implementation guide. The implementation guide says that the first value refers to a code that indicates the Transaction Set Purpose. A value of '00' for BEG 01, indicates that this is a new order. The second value of 'SA' for the BEG 02 element, is the "Purchase Order Type Code". A value of 'SA' indicates Stand Alone. The next value of BEG 03, 'A123456', is the purchaser's purchase order number.

The next statement are two asterisks with no value in between. This is because the next value (BEG 04) is either an "optional" or "deprecated" value. This is where EDI starts to show its complex nature. One would think that if something is deprecated, it should just be deleted. That is not the case with EDI because this could cause major breakages within EDI infrastructures. Major problems would occur if one removed one asterisk - next field would be wrongly interpreted as the preceding, giving way to an avalanche of errors. Instead, EDI deals with deprecated fields by leaving them in and marking them deprecated. The final value, '19980507', is the date in an ISO 8601 format.

The last character in the first line is the segment termination or the pipe symbol ("|"). This indicates that the segment is over and a new line will likely start with the next segment.

1.7 EDI Architecture

EDI is usually deployed in between communication end points between two companies. Each company deploys a server that acts as an end point of gateway for EDI communications. All communications are routed by a special service provider, which gives you access to a "Value Added Network" or VAN. A VAN provider makes sure that all your EDI transactions are routed to the correct recipient and are secure and safe when being transported, as well as providing other valuable services your organization will likely use.



In a standard scenario, a buyer's procurement system generates a signal to the EDI Mapping and Translation Software. A transformation is done and an EDI format message is formed. The message is then passed to the VAN operator who routes it securely and privately to the Seller's EDI mapping and Translation software. The Seller then accepts the message, translates it and places the relevant parts of the message into its back end Order Management System.

A VAN operator usually charges a transactional fee for this service.

1.8 Costs of EDI

Only after looking at some aspects of EDI, it should be clear that TCO (Total Cost of Ownership) is very high. To break even, the savings must be sizable as well. Obviously the amount of savings depends on the number of documents exchanged; large companies that process numerous business documents can offset costs in a relatively short period of time.

It should be stressed that EDI costs consist of creating and maintaining a set of rules, developing and implementing the tools for document management as well as development of server platforms (both software and hardware) that would be capable of handling EDI data interchange. Moreover VAN provider charges are to be added as well as.

On the other hand, small- and medium-sized enterprises (SMEs) find it harder to justify EDI. Indeed, in practice, EDI has developed under the momentum of larger companies. Most SMEs who are using EDI have done so under the pressure of a larger business partner.

Nowadays more SMEs are equipped with PCs and have access to networks, thanks to the Internet. On the business side, the role of SMEs is growing. SMEs have more interaction with large companies to buy or sell goods and services; this results in a greater flow of business documents between them.

There are traditional barriers to entry for EDI, such as large cost, trading volume requirements and proprietary software. Many large enterprises and government organizations have invested greatly in traditional EDI systems over the years; therefore, companies of all sizes need to be able to handle XML and EDI data formats.

However, some of the EDI aspects (e.g. security issues) have been successfully replaced by new, cheaper and usually more effective tools. Therefore, with the introduction of XML, many companies are moving towards XML tools and applications, since they are less complex than EDI and cheaper to operate and support.

1.9 A Word about XML⁷

XML stands for *extensible markup language*. XML is not so much a language as a methodology for marking up data. XML has very strict rules for parsing and its construction of input streams. XML

⁷ More information can be obtained from the W3C site on XML where one may view the actual XML specification.

has the ability to provide metadata document as a set of rules for the structure of each XML document. These metadata documents are called *Schemas* or *Document Type Definitions* (DTD's).

XML is fairly simple and has relatively few rules of grammar to observe. Its syntax is based on a language called *Standardized General Markup Language* (SGML). The concept of XML is to use a "container" for marking up the "content". The container, in XML's case, also has a label or name to tell the reader what the content is. "Element" is denoted by two angle brackets on either side of the element name. Elements usually come in pairs and they contain the content. There are opening elements and closing elements; the closing element uses a "/" symbol right after the first angle bracket.

XML was created to make an industry standard for marking up data. The actual element names used and the structure of the documents are left up to individual design preferences. Business users can design their own set of documents for exchanging business information between end points.

1.10 XML and Business Applications

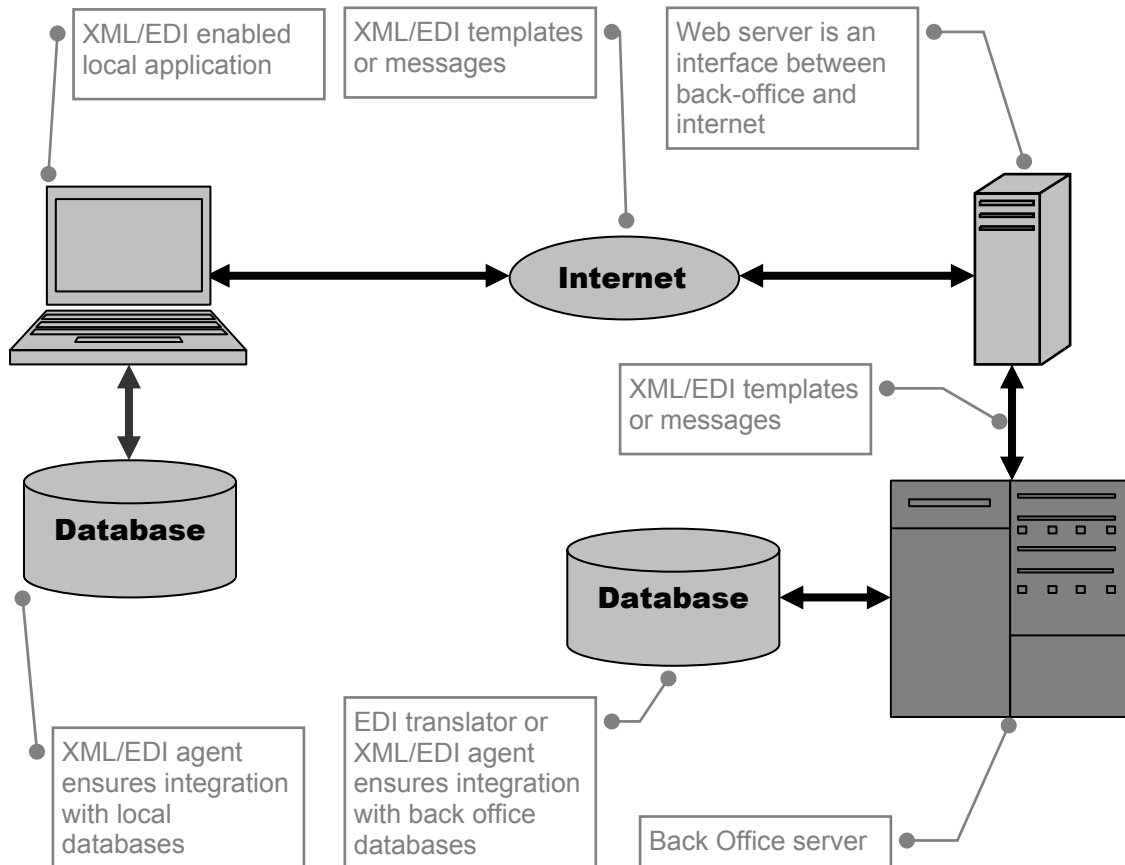
In instances where the XML will only be used within a closed environment, the element names and document structure are not usually given much thought or input from other sources. If, however, it is decided that XML will replace EDI for Business to Business transactions, more consideration has to be given to the design of the XML. What most organizations have done is establish a community of people who will send and receive these messages and agree on the names or elements/attributes and the structure of these messages. Such vertical XML efforts are usually called vocabularies or taxonomies. Examples of vertical vocabularies are:

1. UBL - [Universal Business Language](#)
2. HR-XML - [Human Resources XML](#)
3. xCBL - [Common Business Library](#)
4. XBRL - [eXtensible Business Reporting Language](#)

However it has to be remembered that XML does not provide semantics itself, thus allowing to freely extend sets of vocabularies. In fact, there are over 1,000 different business vocabularies based on XML at the time of this thesis.

1.11 Combining XML and EDI

Most organizations usually start their XML/EDI strategy with a simple pilot project behind their corporate fire wall. This may be done in either one of two main categories of XML deployment models - the *Transactional* model and the *Aggregation* model.



The Transactional XML model is a model that relies on XML messages being sent between two architectural components. These XML transactions are usually built at one end-point, sent to the second end-point, then acted upon. The actions could result in a return message being sent back, or a log file being appended, or a business event being realized.

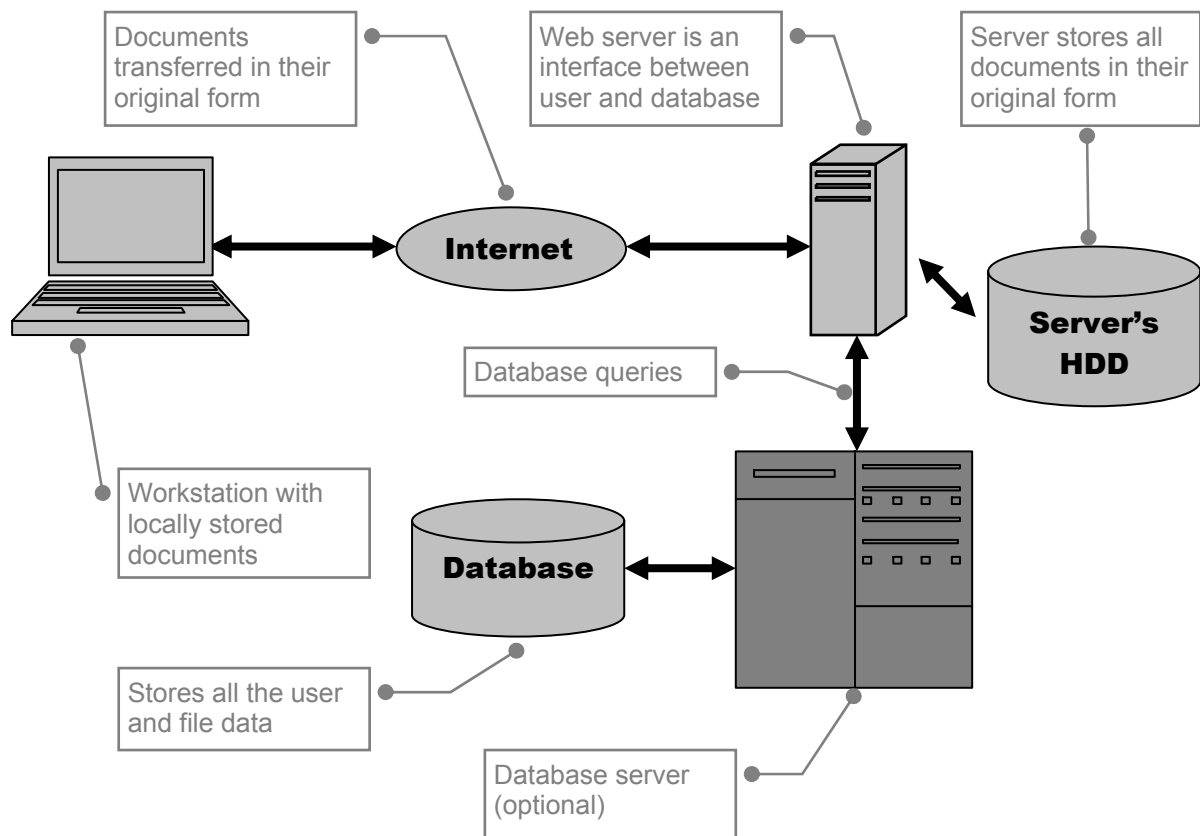
The Aggregation model usually involves aggregating data from many different sources and formats into a single data format, in this case XML. The data aggregation model requires a special piece of software at every data integration point that can transform the data from the native format to XML. Such software functionality is now built into most commercially available software, much the way the ability to export Comma Separated Value (CSV) was built into most spreadsheet programs.

A pilot project might involve accepting incoming EDI transactions and immediately translating them into XML for use by several internal systems. Such a pilot would make economic sense since XML is a universally accepted data format and more and more commercial software is accepting XML as input. XML is also easily transformed into many other types of data such as SQL statements for relational databases, CSV, CDF, or even HTML to view in a browser. Such a project might also

predicate an organization deciding to accept the incoming transactions in XML as well. It is important to ensure that XML can be digested internally before accepting XML as the native format for business transactions with third parties. By using XML as internal data source, one has many advantages that can be leveraged into both short and long term *Returns On Investment (ROI)*.

1.12 WEBDOC – an Alternative

WEBDOC system has been developed to satisfy a specialist end-user requirement. Here the customer wanted to create a fast and scalable document exchange system providing extended security features which would store information (documents) in their original form. Hence the main difference between WEBDOC and EDI is the lack of need to translate documents between the end user and server system. This feature has a serious impact on the system's architecture.



It is very important to notice that the whole system consists of third party Public license based tools which greatly improves the short and long term ROI. Furthermore due to the approach similar to EDI and yet without it's major problem of consistent data storing format we achieve fast and reliable system which can be used in small and large companies and enterprises.

1.13 What is WEBDOC exactly?

WEBDOC is a system for managing, uploading, viewing and downloading documents. It is a typical three tiered architecture project consisting of Linux Server, WWW Apache Server with the PHP script engine and finally a MySQL database. This system is a perfect substitute for the tedious distribution of files using the FTP protocol. It is not only restricted to the documents it can distribute binaries, source code, images, etc.

The system uses cookie-based user verification. It is fully compliant with the user verification systems known from the most OS. Users can be assembled in groups. Each user and each group can be assigned appropriate rights to every file/directory. User rights are stronger than the group rights. User rights are calculated as an effective sum of the rights of all the groups user belongs to and are ANDed with the rights at the user level.

The system supports a normal file system-like management of stored items. It allows creating directories and sub-directories in which files can be placed. It allows the deletion of whole directory trees along with the files contained within. Simultaneously to the process of deletion the process of user rights verification takes place and the former is halted in case the access violence exception occurs. Files are uploaded using HTTP POST method which allows for seamless operation even in the proxy/fire-wall protected networks.

System provides also auditing utilities. It monitors who has logged in, from which computer and what client software the user is equipped with. It allows tracking of possible hack-in attempts as well as user monitoring. The same monitoring is conducted with the files.

WEBDOC system is supported with neat and clean interface. It is easy to navigate and intuitional thus not requiring any special training for the end-user to be able to fully benefit its advantages. It has been designed with the possible future development in mind hence it is well documented and thought of as a framework with enlargement possibilities – an advantage that cannot be underestimated.

Database design meets and complies with the standards for clarity and supports the 3rd normal form. This is essential from the database consistency's point of view. It ensures the system will function without any leftovers that could possibly be cause of future malfunction. The database is also used as a representative for the hierarchical file system structure, which in case of its purely relational architecture is not so common. The hierarchy has been achieved thanks to the self references within tables and the support for recursion in PHP.

1.14 WEBDOC and Hospital Information Systems

In this chapter, hospital information systems (HIS) and their links to WEBDOC will be discussed. A question: "What is a hospital information system?" will be answered along with other issues usually named in conjunction with HIS and computer science in medicine.

1.14.1 Hospital Information Systems⁸

Literature defines HIS in many different ways and presents various views upon them. Some focus upon the functions of a HIS, while others focus upon the technology used. To begin with, the following definition of a HIS will be used: A hospital information system is the information processing and information storage sector of a hospital. Considering this definition, each hospital has a hospital information system. Nearly all people working in a hospital have an enormous need for information which need to be fulfilled in order to achieve high-quality patient care. The hospital management needs up-to-date-information about the costs and services of the hospital. The quality of a hospital information system is also important for the competitiveness of a hospital. Hospital information systems are therefore an important quality and cost factor. They can be seen as the memory and nervous system of a hospital. The subject information processing is also very complex. Nearly all groups and all areas of a hospital depend on the quality of information processing. The amount of information processing is tremendous and should not be underestimated. Additionally, the information needs of the different groups are often based on the same data. Therefore, integrated information processing is necessary. If hospital information systems are not systematically managed and operated, they tend to develop chaotically. This, in turn, leads to negative consequences, such as an increase in costs and a loss of quality. Systematic management and operation can largely contribute to efficient patient care and to lower costs.

In the 19th century, many societies were marked by power-producing industry and industrial production. At the latest by the second half of the 20th century, the idea of communicating and processing data by means of computers and computer networks was already emerging. Today we speak of the 21st century as the century of information technology, or of an *information society*. It is expected to become a century in which informatics will play a key role. Information, bound to a medium of matter and energy, but largely independent of place and time, shall be made available to people at any time and in any place imaginable. Information will find its way to people, not vice versa. Progress in information processing and information technology changes societies. Globalization of labour, as a consequence, is already significantly changing our world.

Nearly all hospital professionals need a vast amount of information. It is essential for the quality of patient care and for the quality of hospital management to fulfil these information needs. When a patient is admitted to a hospital, the physician normally first needs information about the reason of admission and about the anamnesis of the patient. Later, he needs results from clinical, laboratory and radiology examinations, only to mention the most important diagnostic examinations. This information should be available on time, it should be up-to-date and valid. If it is not available on time, or if it is old or even wrong, the quality of patient care is risked. If this causes repetition of examinations or expensive searches for information, the costs of healthcare also increase. Information should be

⁸ The following two chapters have been written basing on the following publications: Armoni, A. (2002), Effective Healthcare Information Systems, Idea Group Publishing and Rada, R. (2002), Information Systems for Health Care Enterprises, HIPAA-IT.com

documented adequately, enabling other professionals taking part in patient care to access them. This also applies for the information needs of nurses.

People working in hospital administration must also be well informed in order to carry out their tasks. They should be informed on time, extensively and receive current information. If the information flow is too slow, then interests are lost, for example, if bills are written days or even weeks after the patients have been dismissed. If information is missing, for example, payable services can therefore not be billed, the hospital's income is reduced. This, in turn, causes a reduction of the hospital's expenditures which could otherwise be used for patient care.

Hospital management also has an enormous information need. Up-to-date information about the costs and services are necessary as a basis for controlling the enterprise. Information about amount and quality of patient care are equally important; for example, about the form and severity of the patients' illnesses, about infections or about complication rates at therapeutic procedures. If this information is not accurate, not on time or incomplete, the hospital cannot be controlled adequately – considering all the risks of management errors. Information is non-material, nevertheless it belongs to the most important goods of a hospital.

An estimated 3 - 5% of the costs of an enterprise, are due to "electronic data processing." This includes computer systems, computer networks and computer based application systems. Both investment costs and regular costs, such as personnel costs, are included in this estimate. In regard to technical progress, this rate may continue to increase. The costs for general information processing are even higher. Some studies have observed that 25% of the a hospital's costs are due to information processing.

According to the "Statistisches Bundesamt"⁹, in 1996, the costs for German hospitals and other German healthcare institutions for prevention and rehabilitation amounted to 49 billion EUR. 1,1 million people worked in these institutions. 15,2 million in-patients were treated. For in-patient care, 593.743 beds in 2.269 hospitals and 189.888 beds in 1.404 health institutions for prevention and rehabilitation were offered. In addition to all this, comes the out-patient treatment which is mainly carried out in the 40 university hospitals. The investment costs, financed primarily from public funds, are not yet included in the mentioned costs.

According to State Statistic Office in Poland in 2001 all the 736 hospitals offered 188234 beds of which 691 were public hospitals offering 185758 beds. In year 2000 there were 85031 doctors¹⁰, 11758 dentists, 22161 pharmacists and 189632 nurses along with 21997 midwives.¹¹

Based on these figures, it becomes obvious that the quality of information processing in hospitals is also a very important cost factor. It even has an enormous significance for the national economy. It is clear that efficient information processing offers vast potential for cost reductions. On the other hand, inefficient information processing will lead to cost increases.

Figuratively speaking, a hospital information system can be seen as the memory and the nervous system of a hospital. The hospital information system is the information processing and

⁹ German Research Association (DFG): Annual Report 1997. DFG, Bonn, 1998. (Deutsche Forschungsgemeinschaft (DFG): Jahresbericht 1997.)

¹⁰ Of all specialties (including GP's)

¹¹ Author of this paper was unfortunately unable to gain access to financial statistics of Polish health care facilities.

storage subsystem of a hospital. It receives processes, stores and presents information. The quality of a hospital information system is essential in order for the hospital, again figuratively, to be able to adequately recognize and store facts, to remember and to act.

1.14.2 WEBDOC in HIS environment

Important issue needing to be discussed in this paper is how does WEBDOC fit in the HIS infrastructure. It has been already stressed that WEBDOC is not a standalone HIS, neither does it fulfil the requirements of Financial Information Systems in a strict sense of this word. However as stressed in the previous chapter WEBDOC system meets all the requirements for the Information Management system. It allows to store and process information. Furthermore it supports a hospital with effective user management system. Moreover it is equipped with sophisticated security policy management system, which proves to be of extreme value in a Hospital environment. One of main requirements all such systems have to meet is data security, required by the law of the country a hospital is situated in. WEBDOC with its user rights management mechanism along with data encryption (SSL) and data security (possible use of EFS) meets all these needs.

What makes WEBDOC application so desired is its flexibility and possibility to instantiate it according to the needs of target Business Unit. It is very easy to imagine every ward having its own server and database containing the documents, with one master server managing and storing and administrating all data. Regarding WEBDOC as a document management system with Internet access one gets a powerful utility to manage and store ALL kind of hospital data. It plays no difference whether these are images, patient documents, bills or audio recordings. As long as it is digital it is WEBDOC compatible. Mechanism of structural data storage very well describes the hierarchy in hospital, simultaneously not being limited to any specific solution and thus allowing virtually infinite reconfiguration and extension possibilities.

WEBDOC is not a dedicated solution for any "physical" implementation. It results in some burden of labour that needs to be performed in order to implement it in a specific environment, however once implemented it proves to be indispensable, resulting in many hours of administrative work saved. Internet access to all the hospital documents over secure connections helps "out of office" employees to gain access to always the newest data.

WEBDOC not being HIS itself is an integral part of it.

2. CHAPTER TWO – THE DATABASE

2.1 Summary

The aim of this chapter is to highlight the process of choosing the most suitable database for the application of the web based document distribution system. The core issue considered is that the system is running thoroughly under the GPL license (except for the development Operating System) and hence the database choice is limited to three main databases fulfilling this criterion. These are SAPDB, PostgreSQL and MySQL.

NOTICE: Please pay special attention to the last chapter of database discussion. It addresses some vital information for the database development. It also contains development-specific data. IT IS STRONGLY RECOMMENDED TO READ THIS CHAPTER BEFORE PERFORMING ANY CHANGES TO DATABASE SYSTEM.

2.2 SAPDB

This is relatively the most advanced of all three databases. It is manufactured under the license of SAP AG and is distributed under the GPL license under the condition that the database will be used independently of the other SAP products. However it has to be stressed that this system due to its professionalism is not as widespread as other databases. SAPDB is designed for the business enterprise, with 24x7 hours per week uptime, scalability and high performance in mind. There are no limitations on database sizes or on the number of users. SAPDB is ACID compliant (fully supports ISO-SQL 92 Standards) and includes all RDBMS and enterprise features expected in an open DBMS such as Views, triggers, foreign keys, constraints of various kinds, stored procedures, versioning, hot backups, etc. This allows SQL applications written in other databases to be easily portable to SAPDB. SAPDB also includes a C/C++ precompiler and interfaces for Perl, Python, and PHP script languages. The system is shipped with the web-based management system and console like environment.

2.3 PostgreSQL

This is an object oriented approach towards creating comprehensive database systems that would answer the market needs. The advantage of this system is that apart from implementing the object oriented database it provides a set of SQL compatible commands for administrating the data. PostgreSQL is an Object-Relational database management system that supports almost all SQL constructs, including subselects, transactions, and user-defined types and functions. It is free to download, use, and modify provided the included copyright notice is included. However this system is designed only to function under the Linux/Unix operating system environment, which bearing in mind the desired portability is a disadvantage. In comparison to other database systems this one requires explicit platform for development and testing. This was the main factor that has crossed PostgreSQL out of the list.

2.4 MySQL

MySQL database is by far the most common and widespread database management system available on market. Thanks to its GPL licensing programme and platform portability it has gained a strong market position. There are several factors that strongly speak for this system and yet in the period of testing similar disadvantages (as far as severity is concerned) occurred.

Main pros of MySQL are, as mentioned above, the common of use and system's widespread. Thanks to this there exist a lot of examples and tutorials regarding the installation, configuration and maintenance of the database. Furthermore besides the tutorials there exist quite a lot of practical applications and ready-to-use schemes that make it possible to learn the tricks and twists of this system.

Another very important feature is that unlike the SAPDB and PostgreSQL, MySQL is most commonly used in the internet applications. It has almost become a twin brother of PHP script language which provides a vast amount of functions to deal with all the aspects of MySQL. Again it has to be mentioned that there exists a lot of examples for usage of PHP in conjunction with MySQL.

Another MySQL advantage is it's availability under the various operating systems. Though it has been developed with the Linux in sight yet its ports to other environments prove to be very successful and apart from the machine architecture dependant features do not differ in any aspect from each other.

Yet another important fact is that there exists a lot of third party tools that allow easy management and administration of MySQL server. Although MySQL provides a text-based interface for carrying out all the operations it is indisputably easier to carry out the same actions in the GUI-based application.

Despite all this advantages MySQL has some very serious drawbacks which however are possible to overcome. One of the main disadvantages is the lack of SQL '92 standard implementation. It has to be remembered that this standard is already 10 years old and a lack of implementation is a very serious problem. Moreover these are not slight mere flaws that are of no extreme importance. These are namely the foundations of modern database management systems.

First of all MySQL does not implement in its default database (MyISAM) type the foreign keys feature. This is truly excluding MySQL from any serious large projects requiring the strong database consistency. Foreign keys are the core feature providing the coherence within the system. Its lack cannot be complemented in any other way since this is the innermost and the most secure way to handle table-key relationships. Foreign keys are however implemented in InnoDB tables; however InnoDB does not provide all the capabilities of the MyISAM tables¹².

Another disadvantage of the lack of implementation of the SQL '92 standard is the lack of transactions, stored procedures and triggers. These mechanisms provide very important way of providing database stability and consistency. They also lessen the code load of the application that is managing the database, since many tasks can be performed by the database itself.

¹² Please take a look at the next chapter

The last disadvantage that occurred during the implementation of the WEBDOC system was the inability of MySQL to interpret the nested select commands. This can be easily worked around however it significantly complicates the course of action that have to be undertaken in order to perform a specific operation.

2.5 Why InnoDB?

It has been decided to use the InnoDB tables because of the following issues. InnoDB tables are transactional: they provide rollback and commit capabilities. InnoDB is the only table type in MySQL which supports foreign key constraints. InnoDB tables are fast, even faster than MyISAM tables in many simple benchmarks. For more information please consult the MySQL benchmarks. InnoDB tables have row level locking: they allow higher concurrency than MyISAM tables which use table level locking, or BDB tables, which use page level locking. High concurrency is reflected in high multi-user performance. InnoDB tables provide an Oracle-style consistent read, also known as multi-versioned concurrency control. *SELECT*s do not need to set any locks and need not interfere with inserts and updates to the same table. No other MySQL table type has this property. There is a true hot backup tool available for InnoDB, which allows to make backups of a running database in background, without setting any locks or disturbing database operation. Multi-versioning also allows dumping tables from database with *SELECT INTO OUTFILE* without setting locks on the tables: the database can keep working while a backup is made. InnoDB tables have automatic crash recovery. One does not need to repair tables if the operating system or the database server crashes, when there is no disk image corruption. InnoDB tables can be any size, also on those operating systems where file size is restricted to < 2 GB.

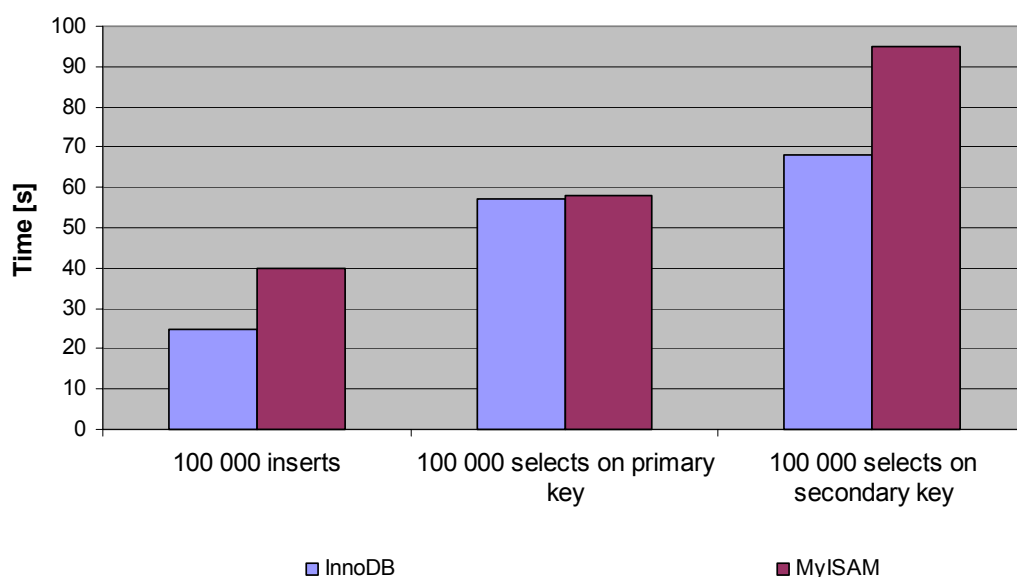


Fig. 4 - InnoDB and MyISAM tables comparison in MySQL¹³

¹³ After www.innodb.com

Very strong position of InnoDB tables is also confirmed by the benchmarks performed on the native MySQL MyISAM tables and InnoDB tables. The results are shown in the chart above. Tests were run on a Linux 2-CPU Xeon 450 MHz machine. The times above are wall clock times.

2.6 Decision

After a long examination of all pros and cons of the above mentioned database systems it has been decided to choose the MySQL database for the system implementation. SAPDB solution turned to be to complex and to robust to tackle with.

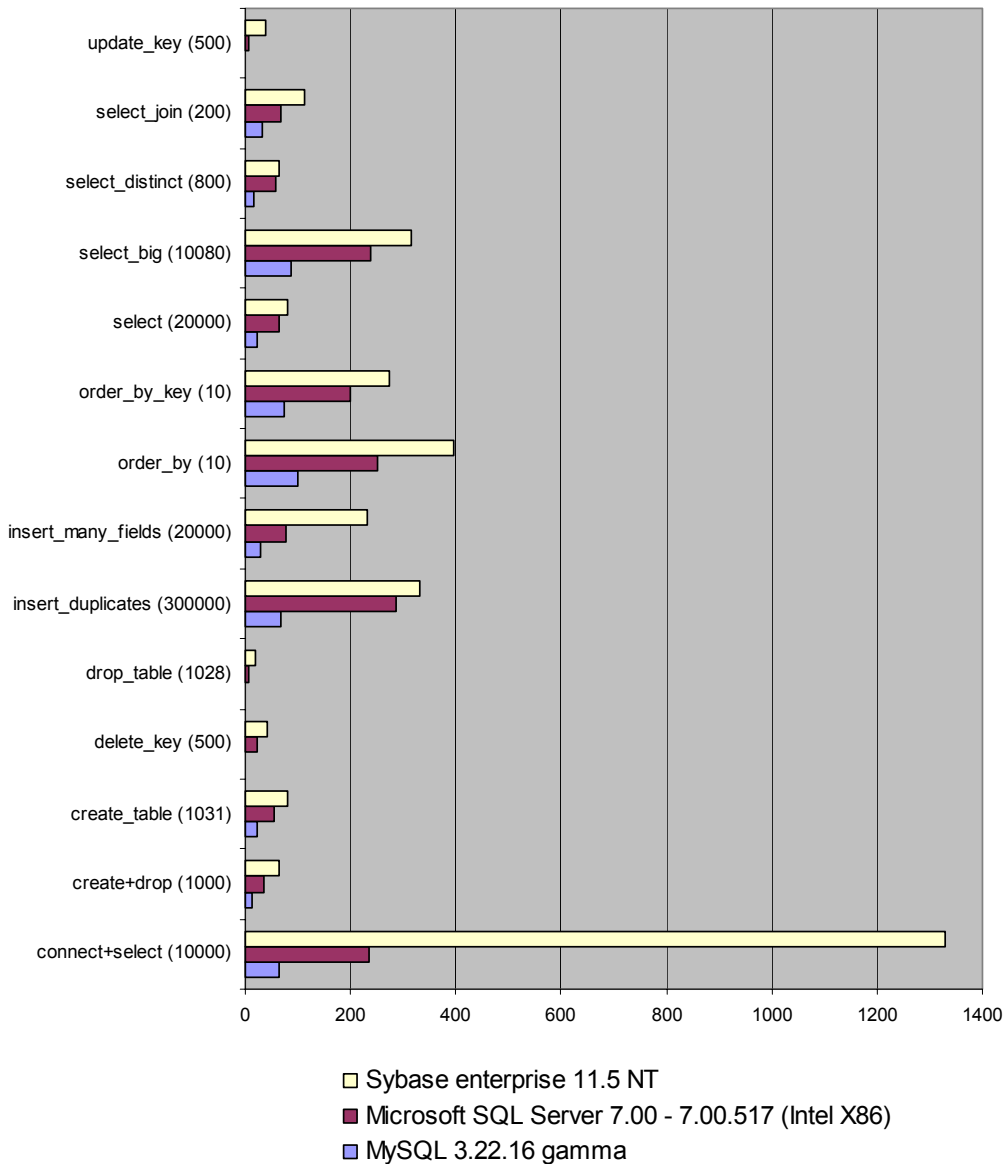


Fig. 5 - Sybase, MSSQL and MySQL comparison chart

It has to be beard in mind that one of the main factors for choosing the database system was also the possibility of future development and the whole project's growth, which inevitably requires

different programmes to develop the system. Hence the choice has been placed on a system that certainly is one of the easiest to understand and master. Very important for the database choose were also the benchmarks comparing the speed of functioning of MySQL database found at the www.mysql.com website. Above is included the benchmark for Sybase, MSSQL and MySQL running on the NT platform. Below is the comparison benchmark of MySQL and Oracle – also conducted on the NT server.

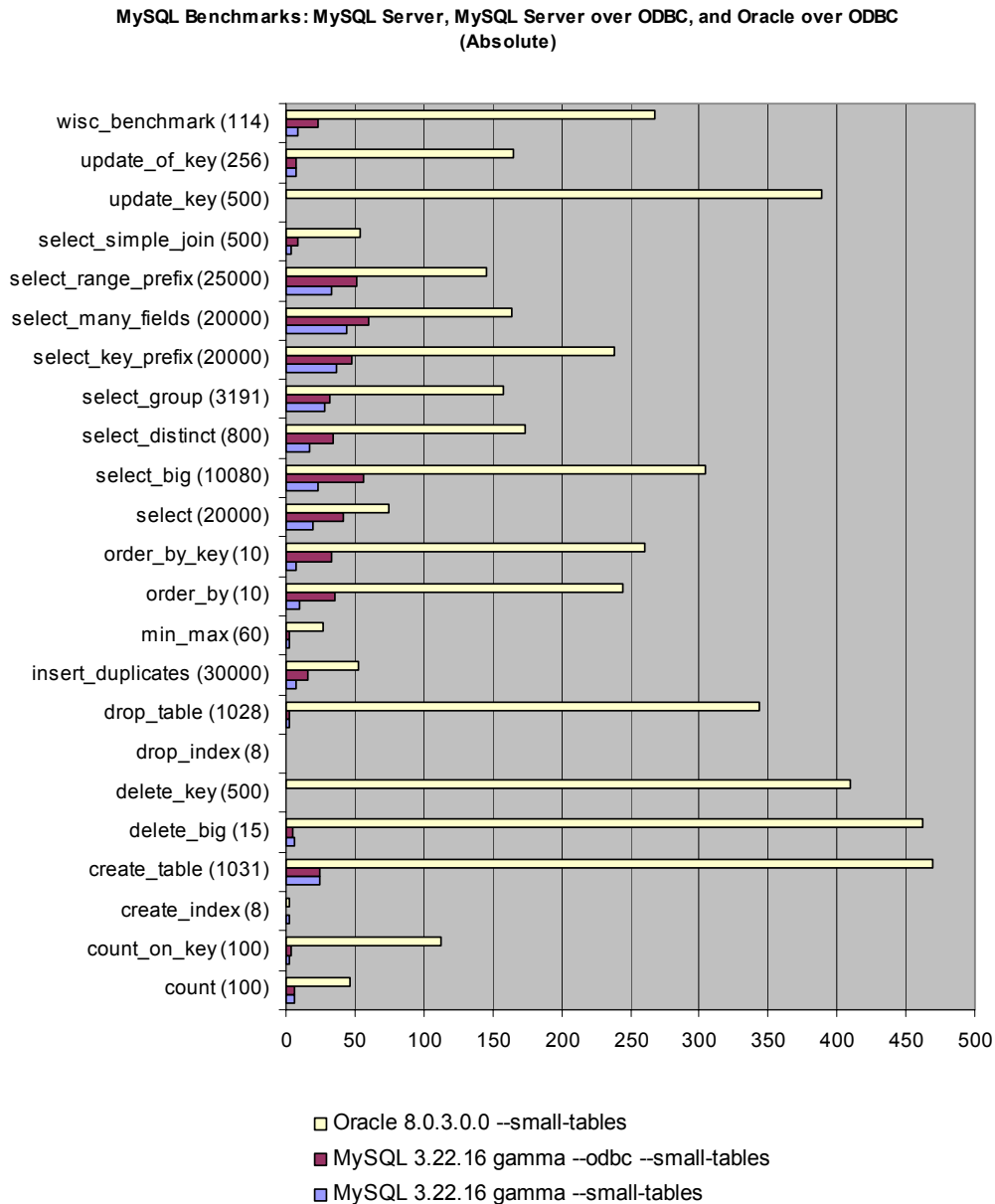


Fig. 6 - MySQL Server, MySQL Server over ODBC, and Oracle over ODBC comparison chart

2.7 Database description

Database system contains data that make a quasi file system with user verification. Since as all the files systems also this one has hierarchical structure of files and directories also the database does have hierarchical bonds between the records representing files. This is a main feature that makes this database a little more difficult to manage. Relational databases were not designed to store such information since for that purpose hierarchical databases were developed. All other database aspects do not differ from the typical databases.

So far the database is in its fourth version which is the one that has been tested on the free standing server. The database consists of 10 tables all of which are of the InnoDB type. The main tables are: `users`, `files`, `groups`, `acl`, `gacl`, and `uglinks`. These tables provide the core of the database system and its functionality. They will be discussed separately.

2.7.1 Database description - `users` table

The `users` table provides the core information about users that are registered within the system. It is a typical information store database. The following code is used for its creation.

```
CREATE TABLE `users` (  
  `id_user` int(10) unsigned NOT NULL auto_increment,  
  `fname` varchar(100) NOT NULL default '',  
  `sname` varchar(100) NOT NULL default '',  
  `login` varchar(100) binary NOT NULL default '',  
  `passwd` varchar(100) binary NOT NULL default '',  
  `email` varchar(100) NOT NULL default '',  
  PRIMARY KEY (`id_user`)  
) TYPE=InnoDB;
```

The main table index – `id_user` field is one of the ways in which users are being differentiated between each other. Another way to do so is to use the `login` field, since all the logins have to be different in order for proper user differentiation. The `fname` and `sname` fields represent correspondingly the first name and the surname of the user. `passwd` field stores the user password and `email` field stores user's email address.

2.7.2 Database description - `files` table

This is the most complicated table of all. Its records contain data about the files and directories. There is no difference between those two for the `is_category` field. Furthermore this table is self-referencing. The following code is used for its creation.

```
CREATE TABLE `files` (  
  `id_file` int(10) unsigned NOT NULL auto_increment,  
  `name` varchar(255) NOT NULL default '',  
  `size` bigint(20) unsigned NOT NULL default '0',  
  `num_acc` int(10) unsigned default '0',  
  `num_dwnl` int(11) default '0',  
  `info` blob,  
  `author_id` int(10) unsigned NOT NULL default '0',  
  `is_category` bool NOT NULL default '0',  
  `parent_cat_id` int(10) unsigned NOT NULL default '0',  
  `mime` varchar(255) default '',  
  PRIMARY KEY (`id_file`)  
) TYPE=InnoDB;
```

The `id_file` field is a mean for identifying a file/category. It is a sort of a pointer to the specified data. Name field contains the filename or category name – e.g. `readme.txt` Field `size` stores the file size, which in case of category is equal to 0. `Num_acc` and `num_dwnl` fields store number of times the specified file has been accessed and downloaded correspondingly. `Info` field contains a brief file description that can be entered whilst uploading the file to the system. `Author_id` stores the `id_user` value of the user which has created, i.e. uploaded the file. The Boolean field `is_category` is a mean of differentiating between files and categories, since in all other aspects they are identical. `Parent_cat_id` is a reference to the `id_file` field of the same table. It provides the means of creating the hierarchical structure of files and directories linked one to another. There exists one root directory with `id_file` equal to 1 and `parent_cat_id` equal to 0. These values are important and should not be changed under any circumstances since they are critical for the correct file system functioning. Field `mime` stores the file MIME type provided by the browser.

2.7.3 Database description - groups table

Groups table stores information about the user groups functioning in the system. For easier administration of the user rights system users can be grouped in groups which can be assigned appropriate rights. Users inherit their rights from the groups. It is important to bear in mind that group rights do sum up – i.e. if user belongs to one group which has the GRANT access to the resource and simultaneously he belongs to the group which has only READ access his effective access right will be GRANT.

```
CREATE TABLE `groups` (  
  `id_group` int(10) UNSIGNED NOT NULL auto_increment,  
  `name` varchar(100) NOT NULL default '',  
  `info` blob,  
  PRIMARY KEY (`id_group`)  
) TYPE=InnoDB;
```

The `id_group` field stores a unique group identifier. The `name` field is however also unique since system will not accept two identically named groups. The `info` field stores a brief group description.

2.7.4 Database description - ACL Table

ACL stands for access control list and it is merely a table storing pairs: user identifier, resource identifier and access type. It is used for gathering the access rights information on the user level. The following code is used for its creation.

```
CREATE TABLE `acl` (  
  `file_id` int(10) unsigned,  
  `user_id` int(10) unsigned,  
  `access_type` int(3) unsigned NOT NULL default '0',  
  INDEX user_idx (`user_id`),  
  INDEX file_idx (`file_id`),  
  FOREIGN KEY (`file_id`) REFERENCES files(`id_file`) ON DELETE  
CASCADE,  
  FOREIGN KEY (`user_id`) REFERENCES users(`id_user`) ON DELETE CASCADE  
) TYPE=InnoDB;
```

The `file_id` field stores the file identifier. `User_id` does similar thing with the system user. The `access_type` field stores one of the four possible access type conneted with the pair mentioned above. They are: GRANT (value 0100b), WRITE (value 0010b), READ (value 0001b) and NO ACCESS (value 0000b)¹⁴. This values are the same for all `access_type` fields in the webdoc database.

¹⁴ The GRANT right allows in addition to WRITE right to set the security rights for the item it corresponds to. The WRITE right allows delete/update/read operations. READ right allows only reading/listing of the file/directory

2.7.5 Database description - GACL Table

GACL stands for Group Access Control List. This table similarly to the previous one contains pairs of groups ids and file ids along with associated with them access types. The following code is used for its creation.

```
CREATE TABLE `gacl` (  
  `file_id` int(10) unsigned,  
  `group_id` int(10) unsigned,  
  `access_type` int(3) unsigned NOT NULL default '0',  
  INDEX group_idx (`group_id`),  
  INDEX file_idx (`file_id`),  
  FOREIGN KEY (`file_id`) REFERENCES files(`id_file`) ON DELETE  
CASCADE,  
  FOREIGN KEY (`group_id`) REFERENCES groups(`id_group`) ON DELETE  
CASCADE  
  ) TYPE=InnoDB;
```

The `file_id` field stores the file identifier. `Group_id` does stores the id of the group whose rights we are modifying. The `access_type` field stores one of the four possible access type conneted with the pair mentioned above. They are: GRANT (value 0100b), WRITE (value 0010b), READ (value 0001b) and NO ACCESS (value 0000b). This values are the same for all `access_type` fields in the WEBDOC database.

2.7.6 Database description - uglinks table

This table stores the connections between users and groups. One group can contain many users and only users. Users can be assigned to multiple groups. The following code is used for creation of `uglinks` table.

```
CREATE TABLE `uglinks` (  
  `user_id` int(10) unsigned,  
  `group_id` int(10) unsigned,  
  INDEX group_idx (`group_id`),  
  INDEX user_idx (`user_id`),  
  FOREIGN KEY (`group_id`) REFERENCES groups(`id_group`) ON DELETE  
CASCADE,  
  FOREIGN KEY (`user_id`) REFERENCES users(`id_user`) ON DELETE CASCADE  
  ) TYPE=InnoDB;
```

User_id field stores user id. Similarly the group_id field stores the id of a group user mentioned above belongs to. Both values are the foreign keys of this table.

2.7.7 Database description - userhistory table

This table has been created in order to provide a sort of an audit on the users logging into the system. It is well known that monitoring the basic user activities (e.g. logging in and out) can provide a powerful tool for user management and improve the overall database/system security. One of the examples could be the deletion of the users that have not logged since a specified date. The following code has been used for table creation.

```
CREATE TABLE `userhistory` (  
  `user_id` int(10) unsigned,  
  `last_ip` varchar(30) default '',  
  `date` datetime default '0000-00-00 00:00:00',  
  `browser_info` varchar(100) default '',  
  INDEX user_idx (`user_id`),  
  FOREIGN KEY (`user_id`) REFERENCES users(`id_user`) ON DELETE CASCADE  
) TYPE=InnoDB;
```

User_id field stores user id. The last_ip field stores the IP address of a machine that user has most logged in. Date field stores the date of logging. Browser_info stores the client's machine specific information (like operating system version).

2.7.8 Database description - filehistory table

This table stores the history of a file – i.e. the list of changes. Each entry (record) stores information about the modifier (his id) and modification date. Optionally a short comment can be provided. The following code has been used for table creation.

```
CREATE TABLE `filehistory` (  
  `file_id` int(10) UNSIGNED,  
  `modifier_id` int(10) UNSIGNED,  
  `date` datetime NOT NULL default '0000-00-00 00:00:00',  
  `info` blob NOT NULL,  
  INDEX file_idx (`file_id`),  
  INDEX modifier_idx (`modifier_id`),  
  FOREIGN KEY (`file_id`) REFERENCES files(`id_file`) ON DELETE  
CASCADE,
```

```
FOREIGN KEY (`modifier_id`) REFERENCES users(`id_user`) ON DELETE
CASCADE
) TYPE=InnoDB;
```

Field `file_id` identifies a file which is being described. `Modifier_id` is an identifier of the user who has done the change to the file's status. `Date` is the date of change/event. `Info filed` provides a brief description for the change occurred.

2.7.9 Database description - keywords table

This table provides a pair of `file_id` and the keywords used to describe this file by an author. The following code has been used for table creation.

```
CREATE TABLE `keywords` (
  `file_id` int(10) UNSIGNED,
  `keywords` text NOT NULL,
  INDEX file_idx (`file_id`),
  FOREIGN KEY (`file_id`) REFERENCES files(`id_file`) ON DELETE
CASCADE
) TYPE=InnoDB;
```

`File_id` represents the file id. `Keywords` field contains all the keywords used to describe a single file. Currently this table is not being used in the system due to lack of the search functionality implementation.

2.7.10 Database description – emailnot table

This table provides the pairs `user_id` and `file_id`. Email notification functioning purpose is quite simple. Whenever any sort of change is applied to the specified file a user who has his id connected with this file in `emailnot` table receives an email informing him about the change. It is supposed that this table will contain only entries for directories rather than single files. The table creation code is as follows.

```
CREATE TABLE `emailnot` (
  `user_id` int(10) unsigned,
  `file_id` int(10) unsigned,
  INDEX user_idx (`user_id`),
  INDEX file_idx (`file_id`),
  FOREIGN KEY (`user_id`) REFERENCES users(`id_user`) ON DELETE
CASCADE,
```

```
FOREIGN KEY (`file_id`) REFERENCES files(`id_file`) ON DELETE CASCADE
) TYPE=InnoDB;
```

As mentioned above the `user_id` field stands for a user id and the `file_id` field stands for a file id.

2.8 Database specific data

There are several issues that are to be discussed whilst concerning this database. These issues concern both data and database design. They have also wider influence, regarding the PHP and the system as a whole. They will be discussed in several points.

- There always has to exist a root directory in the files table. Hence the possibility to delete it from the PHP level has been disabled
- There always should exist administrator user. Hence the possibility to delete it from the PHP level has been disabled
- Administrator user should always bear the `id_user` that is equal to 1. This value has been hardcoded and shall under no circumstance be changed
- A mechanism in the php has been provided to ALWAYS add a GRANT right for the administrator to the newly created file/directory
- `Email_not` table is not being currently used in the PHP code nor is being used the functionality it provides
- `Keywords` table is not being currently used in the PHP code nor is being used the functionality it provides
- User that connects with the database is called `php` with the password of `php`
- Database table creation scripts and scripts for entering the data to the database are stored in the main project's directory.
- If you want to fiddle with the database it is **STRONGLY** recommended to print out the database diagram and keep it up to date.

2.9 Database diagram

The below listed diagram represents the connections between the tables in the database. FK stands for Foreign Key whilst PK stands for Primary Key. Field names are given along with the datatype description. Arrows represent the 1:N cardinality.

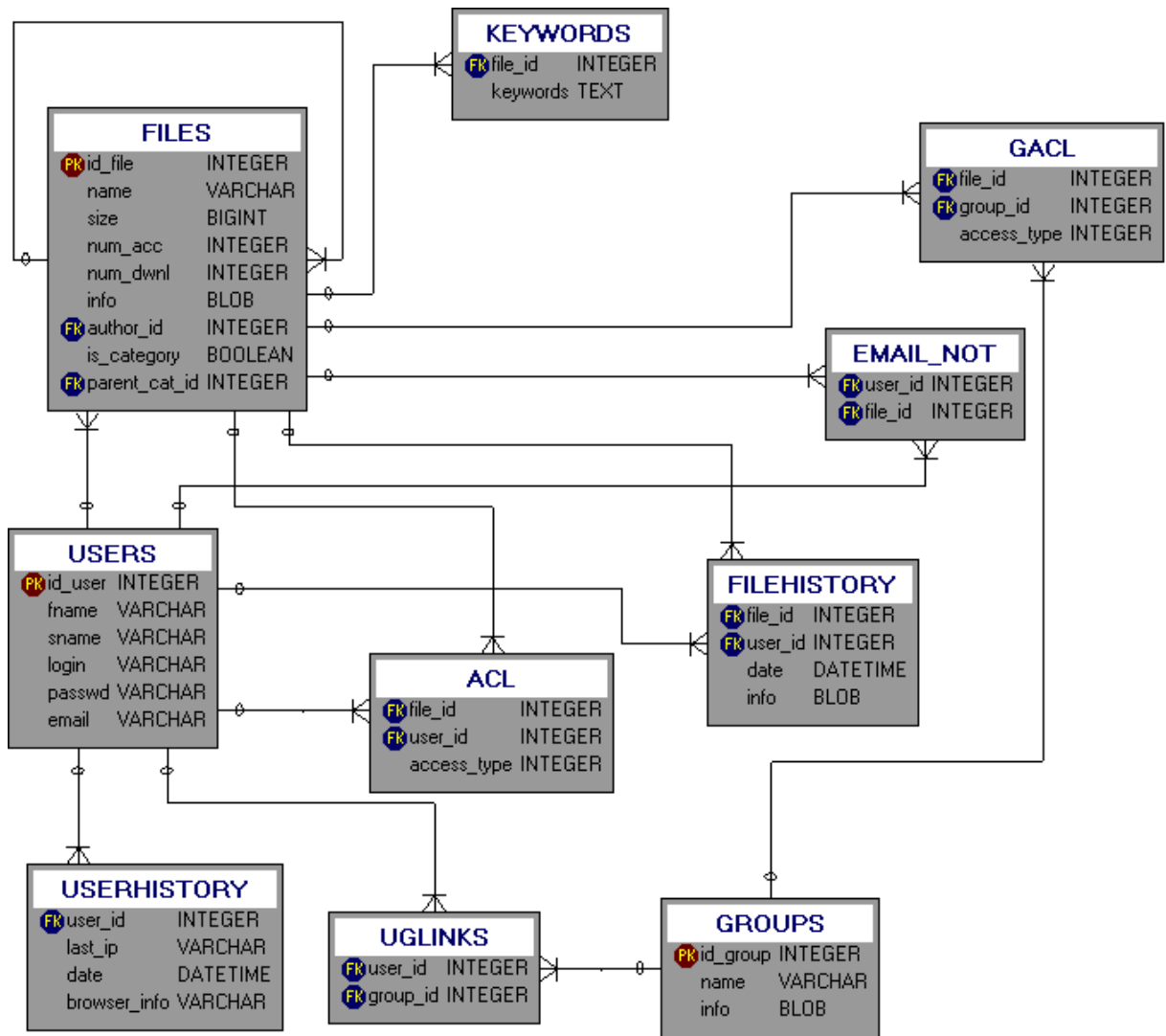


Fig. 7 - WEBDOC database diagram - version 4

3. CHAPTER THREE – THE SYSTEM

3.1 System highlights

The system starts functioning as soon as all the necessary components are installed on the target computer(s)¹⁵. Upon connecting to the system user is presented with the login screen, where he is prompted to input his valid system username and login.



Fig. 8 - WEBDOC main login page

If user provides an incorrect login or he misspells his password he is presented with the appropriate error message along with meaningful help message highlighting the fact that the passwords and user names are case sensitive. This seems to be the most common source for login errors.

At this point it is worth mentioning that user interface is striking as a generic one – meaning it is not dedicated or focused at the target company. It rather focuses on the system itself. Such idea has been applied due to the amount of the system implementations. So far it is running at the State Psychiatric Hospital in Rybnik and at the FH Aalen. It would be extremely hard to provide a “personalized” interface for both of the institutions. It would practically involve rebuilding the whole

¹⁵ System can be installed on many computers in a sense that a different server takes care of the database and a different one servers as the Internet/Intranet Web server. Current Web server is Apache 2.0.43 for Win32. The PHP Hypertext Preprocessor version is 4.3.1


```
else
{
    $ip=getenv("REMOTE_ADDR");
}
```

This method of retrieving the user IP ensures that even if his computer is connecting from behind the proxy server the correct IP address will be logged. It is very common nowadays for large companies to use proxy servers.

Apart from the user agent provided information the main screen contains the Logout button and help button. Moreover user is presented with three main buttons for switching between files, users and groups screens. The layout of the buttons (their highlighting) is realised in PHP. The following code illustrates the principle of choosing an active button:

```
<td width="120">
    <div id="inhalt">
        
    </div>
</td>
```

The function `$fc->getContentsButton()` inserts an appropriate text corresponding to the path to the suitable image file. The purpose and functioning of these buttons will be discussed later in that chapter.

On the main data screen user is among others presented with the option to search for files in the WEBDOC system. Possible search criteria are the filename and the author name. The search functionality does not include the folders in it's results. This has been provided for the result clarity.

After clicking on any of the files stored in the system user has access to the following options: view file, download file, update file, file security, rename file and delete file¹⁹. These options allow easy and seamless file management. Apart from the aforementioned options in the data field user is presented with the full path and file name. Below he can find the file description (entered upon file upload/rename/update to the system) and file MIME type. Under the file MIME type there are two lines with numbers of file open and download operations – this is useful information for file author allowing him to get insight in file usage frequency.

From that place on till the bottom vertical delimiter user can trace the file history in the system. He is always presented with the name and email of the user who is responsible for the action described in the file history. On the left hand side he can find a date and time stamp of the operation. Finally a short description follows. It identifies the operation conducted upon a file.

As already it was briefly mentioned before at the top of the data field there are three buttons which are used to switch between the file view, the users view and the groups view. WEBDOC System apart from serving as a sort of visual FTP server provides a comprehensive and powerful security and access right management system. These two latter views are responsible for managing security,

¹⁹ Please refer to the Figure 8 in this chapter

groups and user policies in the system. It can be said that the WEBDOC System has two main interfaces – the file interface and the groups/users interface.

The groups and users screens though separate in the system are very similar and thus will be discussed together. At this point it is important to stress that these two views are only accessible for the users belonging to the “Administrators” group.

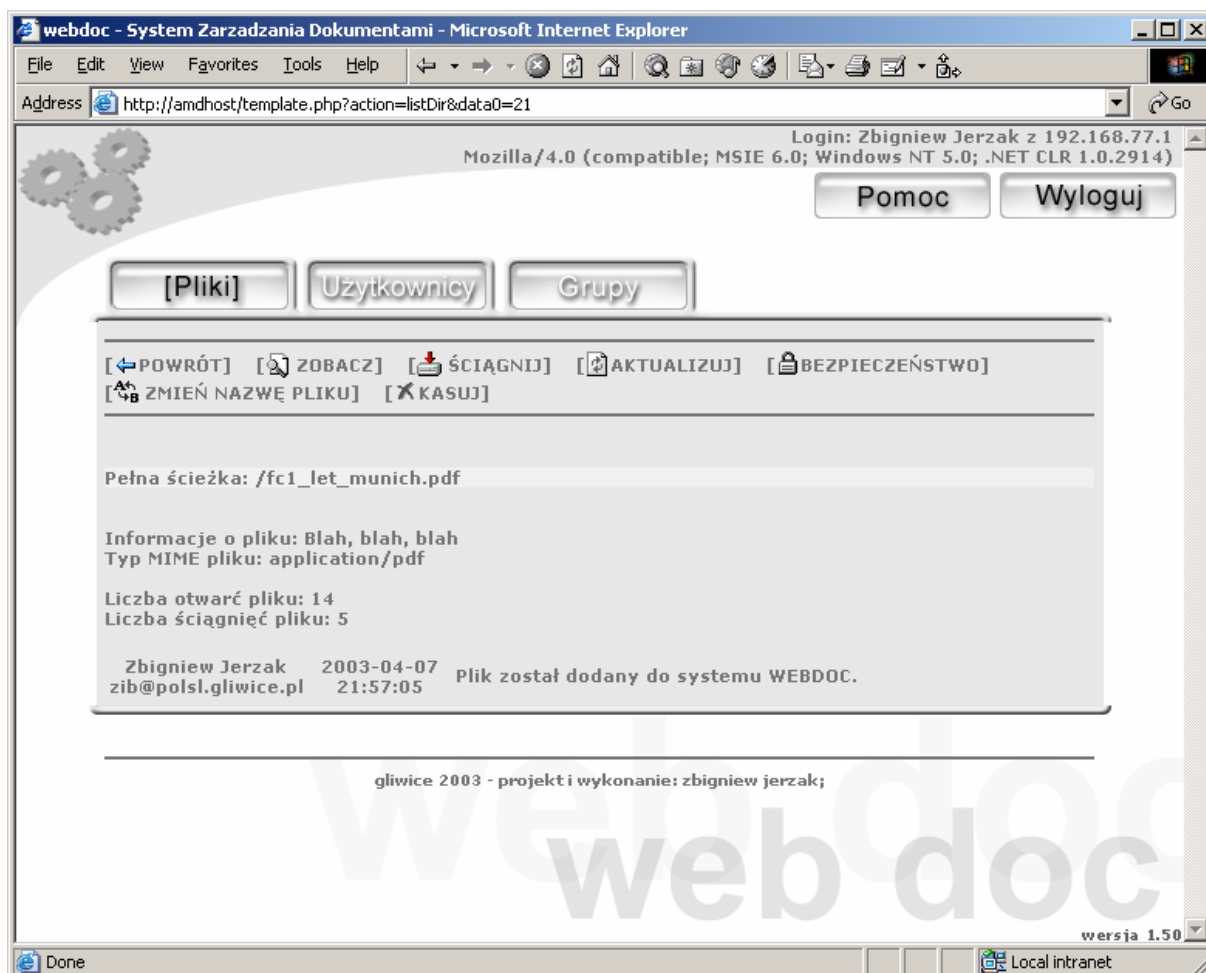


Fig. 9 - File details screen of the WEBDOC System

As one can clearly see in Figure 7 further in this chapter the users screen is very clear and easy to understand. Administrator is provided with a list of all users in the system²⁰, with brief characteristics listed on the same line. As it is clearly visible Administrator is presented with full user name, his login name and his email. It has to be stressed that on this screen there is no division into groups.

From this screen it is possible to either click on the user and enter the user information page or to click on the user's email and instantly send him mail message²¹. User information page contains the full user name and his email address. For security reasons administrator can also have insight to the

²⁰ In case of groups screen administrator is provided with a list of all the users' groups present in the system along with their brief description

²¹ In case of clicking on a group Administrator is presented with group information screen containing a list of all the users belonging to the specified group among with the data similar to the one contained in the users screen page.

details of user's last login. This provides Administrator with information about the user's last IP²² and operating system. It can prove to be very useful in case of any break in attempts. At the bottom of the screen one can find a list of groups the selected user belongs to. Clicking on a group will result in being taken to the group information screen.

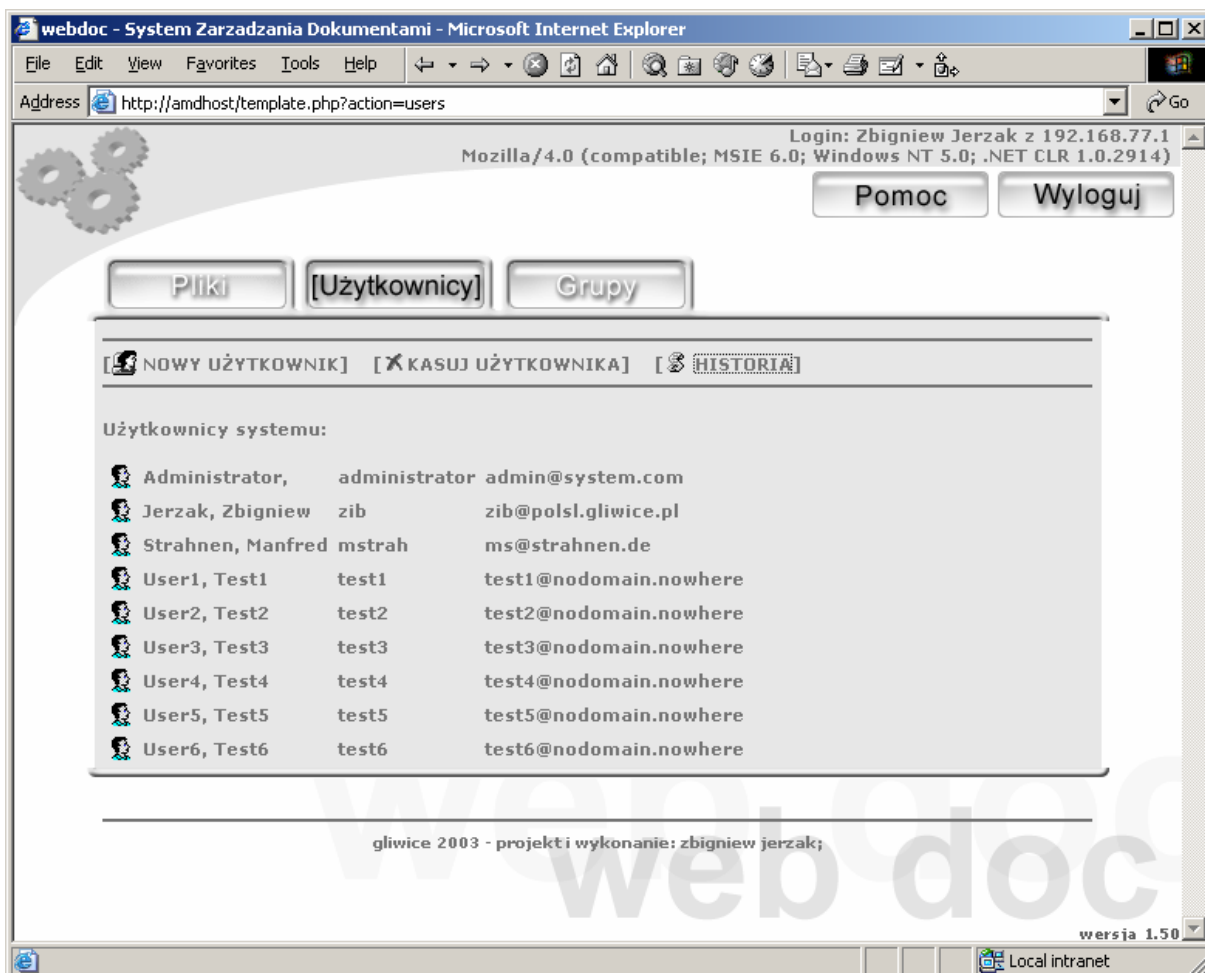


Fig. 10 - Users screen - provides access for administrators to user management functionality

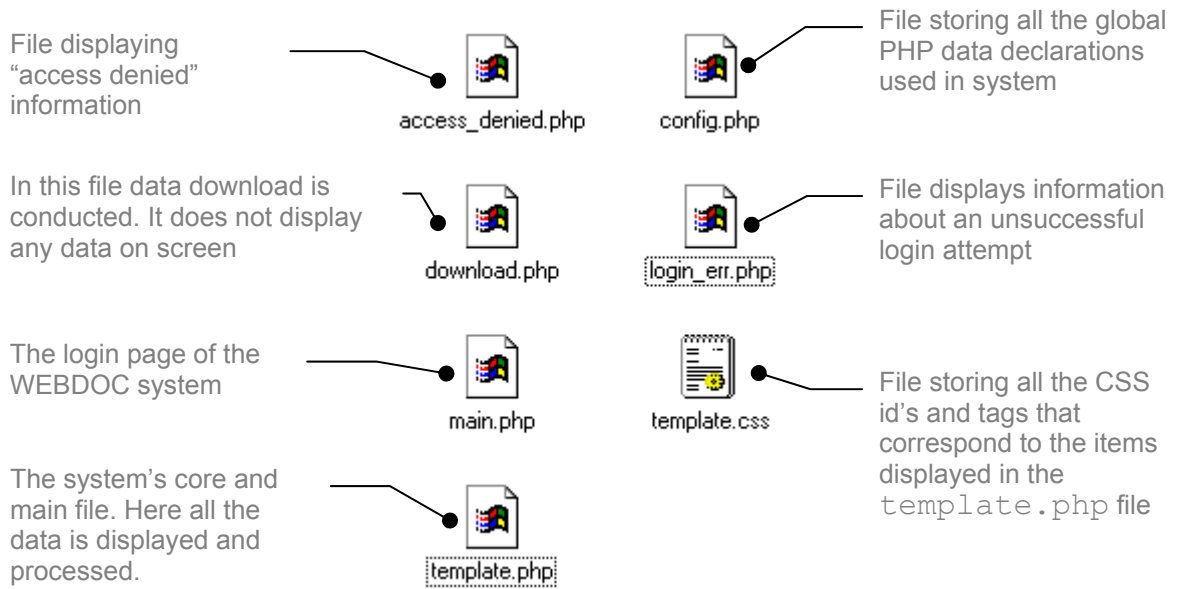
There is one more feature strictly connected with the system's functionality. Namely the user's screen has a history option. It lists a list of all the logins that were made to the system. The list can be sorted either by user ID or by user name or by date. This useful tool allows tracking of "dead users" as well as monitoring the database and system access.

Groups screen is very similar in functionality and layout to the users screen hence it is not discussed here – please refer to the footnotes for some details and more insight.

²² This function is similar to the LAST command in the UNIX/Linux Operating Systems

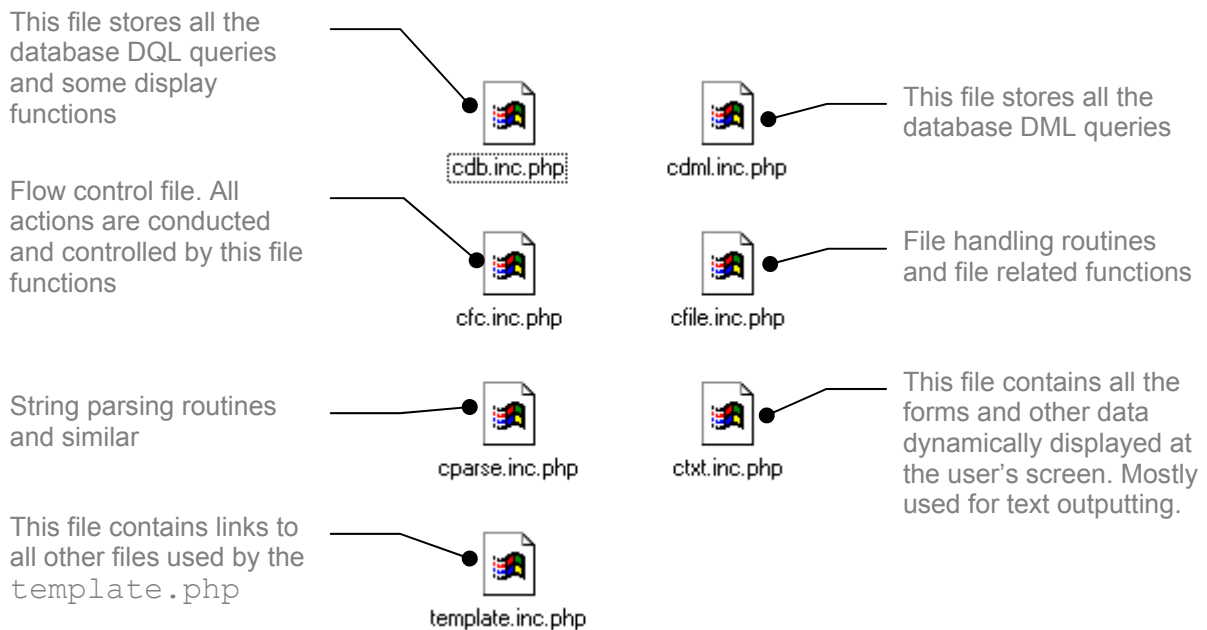
3.2 PHP and HTML Pages

The system consists of the following PHP/HTML pages stored in the WWW root directory:



As shown above the whole system consists of relatively small number of data. The cause for that is that all the data that is being displayed is generated via the PHP code and embedded into the PHP page. In fact it is only one instruction that displays everything. Almost all the code which is responsible for the actions undertaken by the system is stored in the files in `./inc` directory.

The following files are stored in the `./inc` directory:



Every file contains one class of the PHP code. Author has decided to use the object-oriented approach towards creating the system. Object oriented programming is becoming a standard of modern large-scale projects. It provides an easy and clear approach towards coding. Data encapsulation – main feature of OOP provides a powerful toy in fight with hundreds of lines of source code.

Classes created in the programming process correspond with the logical layout of the project. Namely they have been divided in accordance with their functionality, not with the database tables' layout. The second approach could be used however due to the clarity of code and code navigation I have decided to use the first solution. Project contains the following classes:

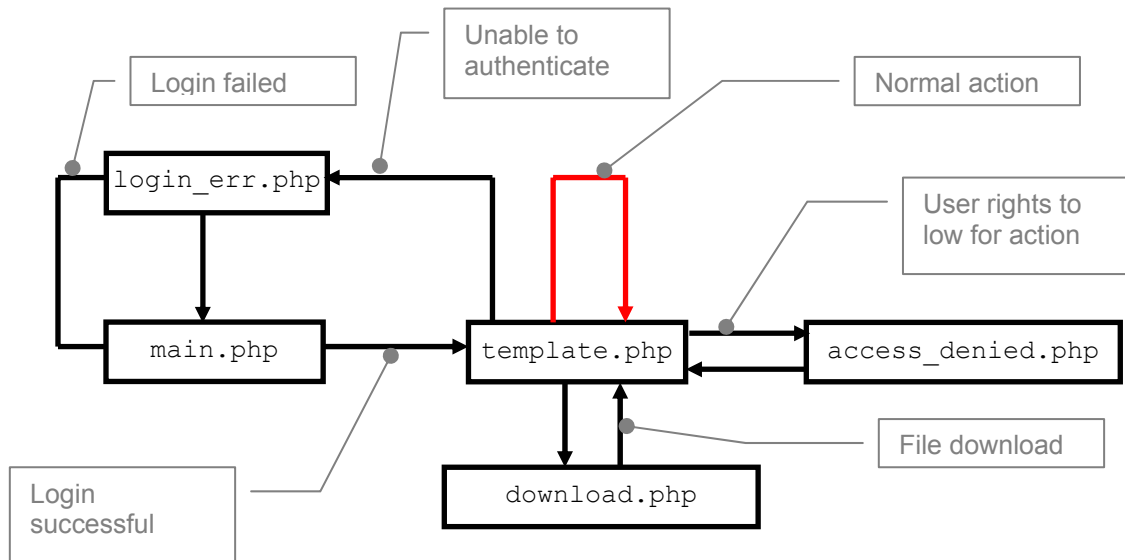
- `CDB` – the database information retrieval class. It performs all the database operations involving the usage of `SELECT` SQL statement
- `CFlowControl` – the flow control class. It receives the form parameters and depending on them performs the appropriate action – it is used to “distribute” work to another classes
- `CDML` – the database information insert- and update class. It performs all the operations involving the usage of `INSERT` or `UPADTE` SQL statements.
- `CParse` – class responsible for parsing the data user has entered into form. It ensures that the data passed to other classes is properly formatted.
- `CTXT` – class dynamically displaying at the user's screen the appropriate form and headings.

- `CFile` – class responsible for file handling and manipulation.

3.3 How to navigate files

There is no point in describing every function and its purpose at the moment. Anyone interested may take a look into the source files and he will find a well documented code that will allow him to easily understand functions' parameters and output values.

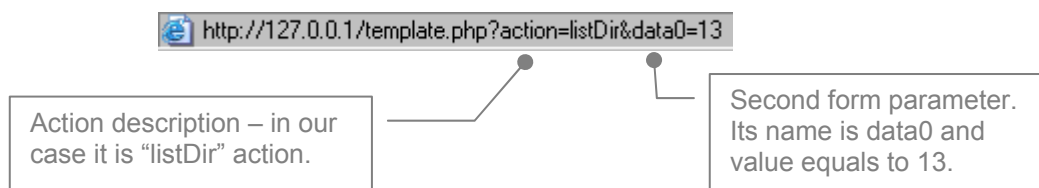
However I think that for anyone willing to develop this system further either by optimisation or functionality addition it would be of great value to highlight the foundation of system functioning. First of all the two main files are `main.php` and `template.php`. The first one contains the login form. The second one thanks to dynamic generation of content is used to display all the data. The work principle is shown in the following graph.



It is clearly visible that thanks to the limited number of files we achieve clarity of functioning. As seen on the graph the core file is the `template.php`. The topmost arrow (red coloured) labelled "Normal action" describes a normal action of the system. There are a few additional files which provide extra functionality. All the arrows stand for data transmission between forms of the project.

So how does it work? When user clicks any link in the `template.php` file he sends a form containing data to the `template.php` file itself. At the very beginning a method from the `CFlowControl` class is called. This method is `$CFlowControl::setAction($action)` its only parameter is the `$action` value. This value is being passed by every form in the project. It tells the `CFlowControl` class what action is supposed to be performed with the data contained in the other form fields.

Usually a typical form data fields sent to the `template.php` file looks like this:



This values (`action=listDir` and `data0=13`) are passed from the `template.php` file to itself. Depending on the `$action` parameter value method `$CFlowControl::setAction($action)` performs initial operations. This usually includes displaying the appropriate images, controlling the access rights and setting initial internal data. Then second method from the `CFlowControl` class is called – this is `$CFlowControl::evalAction($action)`. This method is responsible for displaying appropriate data. A good example is to show how does a blank page `template.php` would look like if neither the `$CFlowControl::evalAction($action)` nor any other method would be called:

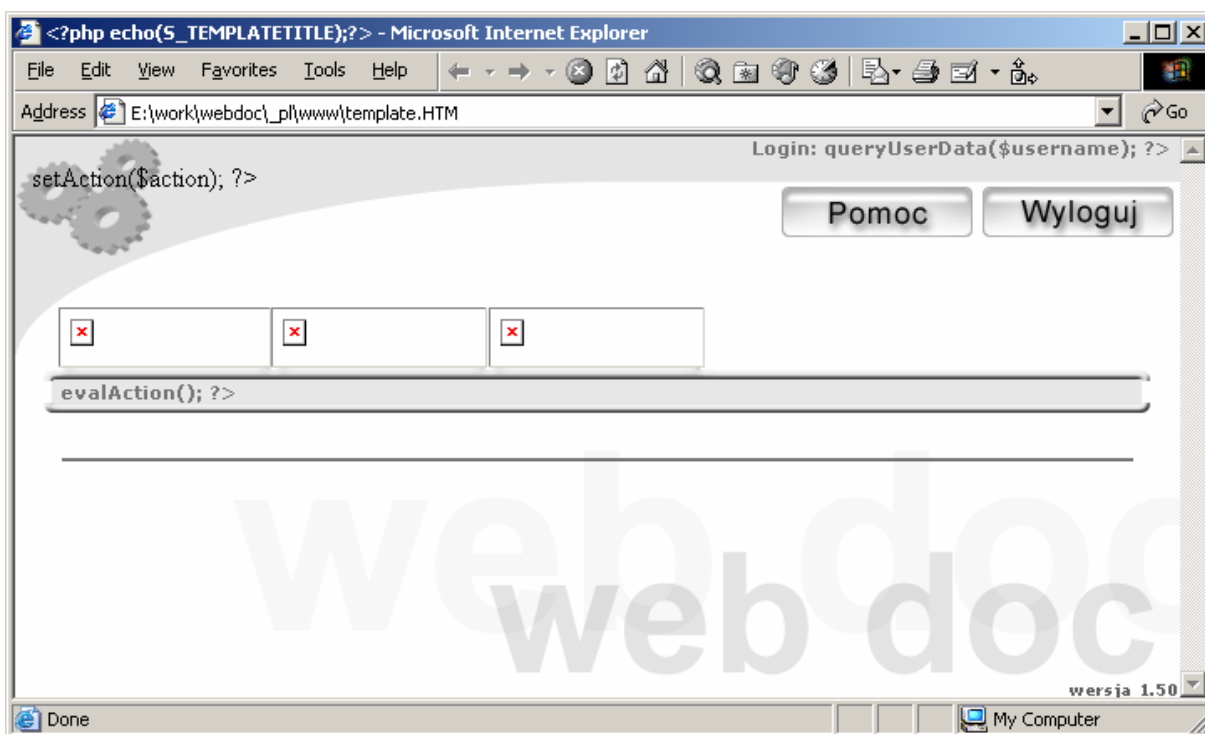


Fig. 11 - Main WEBDOC window without the PHP engine started - the layout of the `template.php` page coded in pure HTML

It is clearly visible that this page is only a template, and cannot function standalone. Only with the `$CFlowControl::evalAction($action)` method it can contain meaningful data. Hence its name. There is also another method visible in the upper right corner of the screen – it is

responsible for displaying the user statistics – where did he log from and what are his browsing tools/operating system. Another method is used to determine the buttons to be displayed above the data field. Here as it is clearly visible this method has not been called and hence the buttons are not visible – compare the screenshot on the next page.

For comparison reasons the following image shows the normal output of the system:

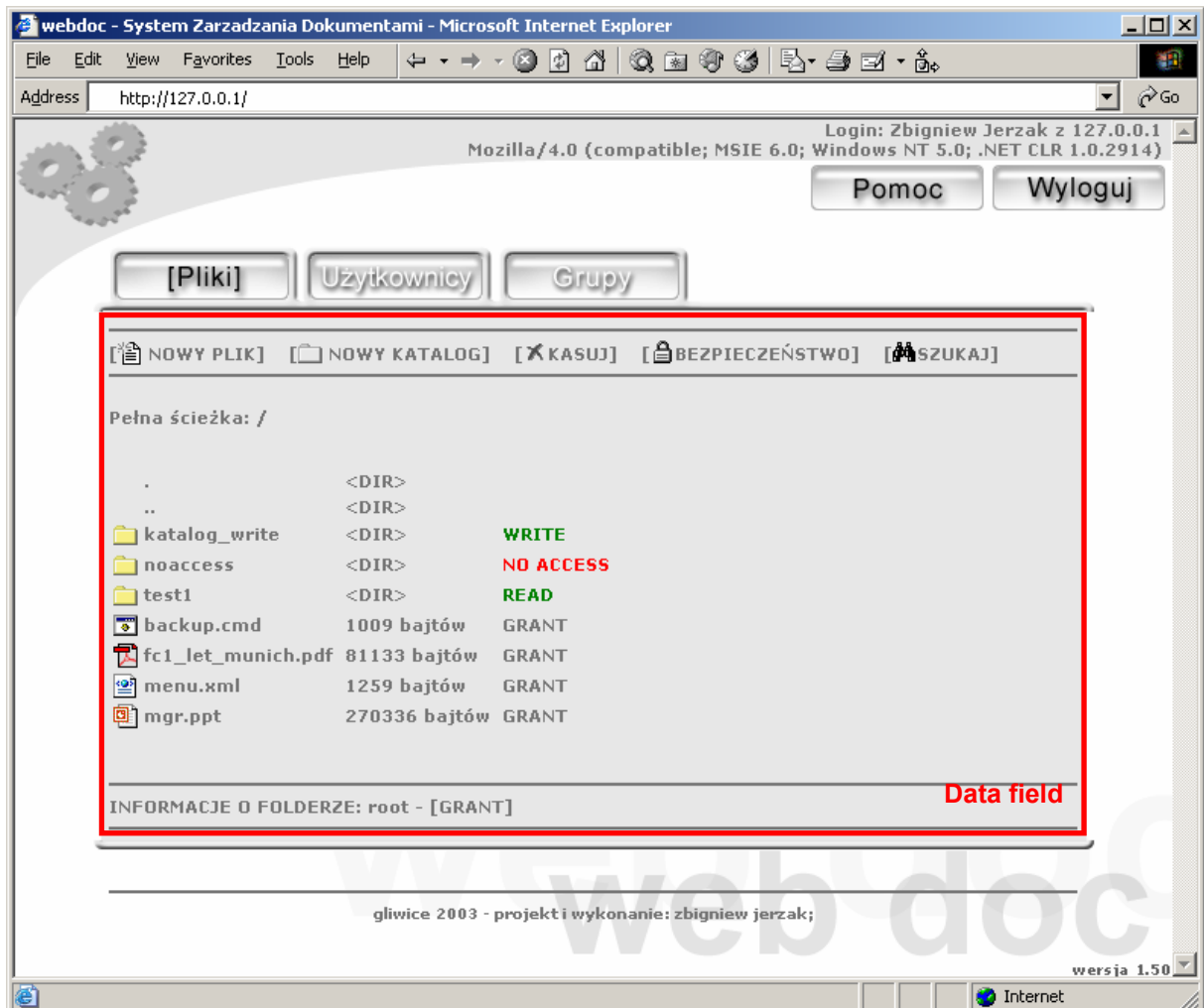


Fig. 12 - Main WEBDOC window with data field

As seen above the `$CFlowControl::evalAction($action)` method provided this time a list of files available in the root directory of the system. They are displayed in the red indicated data field. Also the function responsible for displaying the statistics and information about the logged user is functioning, as well as the one for displaying the buttons.

This approach towards creating the site – i.e. as little files as it is possible is showing it's advantage. It would be extremely difficult to navigate a dozen of pages each calling different functions. Moreover providing some of the functions would be stored within files itself and some in the include files it would result in a chaos, hard to navigate and understand. With this approach (all code in separate files, each representing one class) it is very easy to navigate the project. Moreover, one main

file which displays all data is very similar to the explorer window of either Windows Explorer or Linux Konqueror.

It is also worth mentioning that the `template.php` page does not use frames at all. It has been decided that the frame-free approach is more clear and understandable to the end user. Also the coder does not have to bother with multiple files.

3.4 How to navigate code

Taking under consideration the techniques explained in the previous chapter it is understandable that the amount of code is significant. The principle of the project's functioning has already been explained. Now it will be explained how to find oneself in the code. It is one of the common problems in interpreting the code – how to “bind” what we see on screen with the appropriate lines of the code. In order to solve this problem we have to take a closer look at the `cfc.inc.php` file and contained within `CFlowControl` class.

As it has been mentioned before, each form within the project contains an “action” field which determines what type of action is expected to be carried out on the provided data. This action has a reflection in the `CFlowControl` class. For example let's suppose we receive a login form – it has a field named `action`, which value is equal to `login`. It also contains fields `username` and `password` that store username and password entered by user. When we press the *Anmelden* button we are transferred to the `template.php` page and the contents of the root directory are displayed. A question is – what has happened at the PHP level.

First we have to locate the login action in the `cfc.inc.php` file. The easiest way is to search for the `__LOGIN` variable. This is a text constant containing the text “login”. It is used for all the comparisons within the `$CFlowControl` class. Its first occurrence looks like this:

```
var $_LOGIN;
```

This is the definition of the variable. So knowing it exists we search further:

```
$this->__LOGIN = "login";
```

This is the place where the text “login” is stored in our constant. Now we may seek further into the file. Now we find the following occurrence within the `setAction($action_type)` method:

```
if ($this->action==$this->__LOGIN)
{
    //choose which button to highlight
    $this->contentsButton = "img/dateien.gif";
    //store unname and passwd in cookies first time login
```

```

        $this->setValidateCookies();
//validate user from login page
        $mydb->validateUID($username, $password);
//we set this for root
        $this->currentDirID = 1;
//update the userhistory information
        $dml->updateUserHistory($mydb->uid);
    }

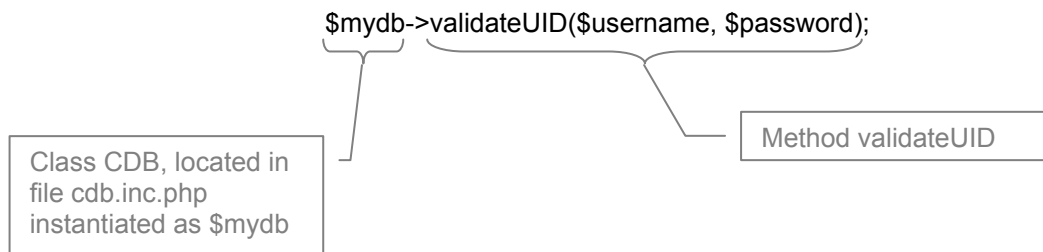
```

Now we know that we have found a piece of code which will be executed when we pass a form with the action field value set to login to the `template.php` file. It is worth reminding that the `setAction($action_type)` method is called from the `template.php` file BEFORE any other action is taken. All the action is clearly documented by the meaningful comments.

All the class definition names are beginning with the C letter, which stands for the word class. For the `CFlowControl` class the convention is **C**(lass)**FlowControl**. The name of the instance of the class is simply missing the C standing for the word class. Hence we declare it like this:

```
$fc=new CFlowControl;
```

If we want to know what does a specific function do it is enough to locate the class it is defined in and find the function. The example might be:



When we proceede searching for the occurrence of the `_LOGIN` text within the `cfc.inc.php` file we find it's last occurrence:

```

if ($this->action==$this->_LOGIN || $this->action==$this->_LISTDIR ||
    $this->action==$this->_LISTDIRUP)
{
    $mydb->controlSecurity(1, $this->currentDirID);
    $mydb->listCurrentDir($this->currentDirID);
}

```

This occurrence finds place within the `evalAction()` method. The first call is controlling the access rights to the directory, whilst the second call displays the contents of the current directory in the data field.

With the following approach it is quite easy to find where the exact action that is being performed within the browser window is defined in the code.

The following table will visualise the name of the file in which the class is declared, the name of the class and the name of the class instance. There is only one instance of each class in the whole project.

Class name	Class' instance name	File calss is declared in
CDB	<code>\$mydb</code>	<code>cdb.inc.php</code>
CFlowControl	<code>\$fc</code>	<code>cfc.inc.php</code>
CTXT	<code>\$txt</code>	<code>ctxt.inc.php</code>
CDML	<code>\$dml</code>	<code>cdml.inc.php</code>
CParser	<code>\$parse</code>	<code>cparse.inc.php</code>
CFile	<code>\$file</code>	<code>cfile.inc.php</code>

Table 1 - PHP File and Class summary

4. CHAPTER FOUR – THE SERVER

4.1 Summary

The aim of this chapter is to clarify the WEBDOC system implementation on a working network server. The server platform (as already mentioned earlier) is Linux. The Linux operating system is SuSE Linux 7.2 The server also runs the Apache 1.3.19-48 and PHP 4.0.4; the installed MySQL server version is 4.0.10-0. This chapter also discusses server (both database and WWW) security issues.

4.2 Server security²³

The most important part of the installation on the web server is providing the appropriate security for the document files and the server as a whole. There are several security levels: the apache security layer, the PHP security layer, the MySQL security layer and finally the server security level – i.e. the Linux security level.

At the Linux security level it is important to provide access only to the authorised users. It is important to bear in mind that users with the direct access to the server are able to alter the web server data in any way. They also have access to all the documents available on the server. This drawback could be partially worked around by placing the documents directly in the database. It is not always desirable that the system administrator would be able to access all the web server contents.

In case server will run multiple systems (not only the WEBDOC system) it is important to provide the appropriate FTP users and assign them appropriate rights to the directories. Otherwise some users might have access to the contents they should not be allowed to manipulate.

At the Apache security level it is most important to provide the appropriate rights corresponding with the appropriate directories of the server. One aspect of Apache which is occasionally misunderstood is the feature of default access. That is, unless one takes steps to change it, if the server can find its way to a file through normal URL mapping rules, it can serve it to clients. For instance, considering the following example:

```
# cd /; ln -s / public_html
Accessing http://localhost/~root/
```

This would allow clients to walk through the entire file system. To work around this, it is advised to add the following block to server's configuration²⁴:

```
<Directory />
    Order deny,allow
    Deny from all
```

²³ Some parts of the following chapter stem from the MySQL manual and website (<http://www.mysql.com/>). Some parts stem also from the PHP website (<http://www.php.net/>) and from Apache website (<http://www.apache.org>).

²⁴ Server configuration file is typically stored in the following location: /usr/local/apache/conf/httpd.conf

```
</Directory>
```

This will forbid default access to file system locations. Adding appropriate `<directory>` blocks will allow access only in those areas it is desired. For example:

```
<Directory /usr/users/*/public_html>
    Order deny,allow
    Allow from all
</Directory>
<Directory /usr/local/httpd>
    Order deny,allow
    Allow from all
</Directory>
```

Particular attention must be paid to the interactions of `<location>` and `<directory>` directives; for instance, even if `<Directory />` denies access, a `<Location />` directive might overturn it.

Also one has to be wary of playing games with the `UserDir` directive; setting it to something like `"./"` would have the same effect, for root, as the first example above. Using Apache 1.3 or above, it is strongly recommended that the following line is included in server configuration files:

```
UserDir disabled root
```

At the PHP level one feature of PHP that can be used to enhance security is configuring PHP with `register_globals=off`. By turning off the ability for any user-submitted variable to be injected into PHP code, you can reduce the amount of variable poisoning a potential attacker may inflict. They will have to take the additional time to forge submissions, and internal variables are effectively isolated from user submitted data.

While it does slightly increase the amount of effort required to work with PHP, it has been argued that the benefits far outweigh the effort. This project uses the global variables. It could be advisable to try to switch the whole code so as not to be forced working with the `register_globals=on` set.

One should always carefully examine his code to make sure that any variables being submitted from a web browser are being properly checked, and ask him the following questions:

- Will this script only affect the intended files?
- Can unusual or undesirable data be acted upon?
- Can this script be used in unintended ways?
- Can this be used in conjunction with other scripts in a negative manner?
- Will any transactions be adequately logged?

By adequately asking these questions while writing the script, rather than later, an unfortunate re-write when one needs to increase security can be prevented. By starting out with this mindset, the security of the system won't be guaranteed, but it can help improve it.

One may also want to consider turning off `register_globals`, `magic_quotes`, or other convenience settings which may confuse you as to the validity, source, or value of a given variable. Working with PHP in `error_reporting(E_ALL)` mode can also help warn about variables being used before they are checked or initialized (so one can prevent unusual data from being operated upon).

MySQL Security needs to follow the following security guidelines. When you connect to a MySQL server, you normally should use a password. The password is not transmitted in clear text over the connection, however the encryption algorithm is not very strong, and with some effort a clever attacker can crack the password if he is able to sniff the traffic between the client and the server. If the connection between the client and the server goes through an un-trusted network, you should use an SSH tunnel to encrypt the communication.

It is also very important to assign password to the root user since by default MySQL starts with empty root user password. This accounts for all the users registered in MySQL database. Anyone can log in as any other person as simply as `mysql -u other_user db_name` if `other_user` has no password. It is common behaviour with client/server applications that the client may specify any user name. It is important not to run the MySQL daemon as the UNIX root user. This is very dangerous, because any user with the FILE privilege will be able to create files as root (for example, `~root/.bashrc`).

4.3 Installation guide

Installation of the server is relatively easy. However one has to bear in mind these several steps for the proper functioning. The installation steps are given in chronological order. Please note that although trying to be as specific as possible some steps of the installation may differ due to the implementation of Linux operating system.

4.3.1 Installation guide - Server, Apache, and PHP

- Install the operating system
- Install the Apache server – it usually ships along with the CD on which the operating system is placed. Latest release of Apache web server can also be downloaded from <http://www.apache.org>
- Create a new directory for the WEBDOC system in the `/usr/local/httpd/htdocs`²⁵
- Copy all the WEBDOC files from the “webdoc - www” directory on the CD to the newly created directory on the server
- Adjust the entries in the `httpd.conf` file:
 - if needed add new `Listen x.x.x.x:y` entry
 - set the appropriate value for the `DocumentRoot`
 - add the `main.php` entry to the `DirectoryIndex` key
 - set the appropriate directory access rights – exclude the `./doc` and `./inc` directories from being able to be view or accessed in any other means by the user²⁶
- Adjust the entries in the `php.ini` file:
 - The `safe_mode` variable must be set to `off`
 - In production state disable the error reporting: `display_errors = Off` and `display_startup_errors = On`
 - Switch the use of the global variables on: `register_globals = On`
 - Set the `include_path` value to whatever the WEBDOC directory and WEBDOC `./inc` directory is
 - Turn file uploads on: `file_uploads = On`

²⁵ Location may vary due to implementation differences

²⁶ Example for disallowing access to PHP include files might be (please consult the Apache website – <http://httpd.apache.org> – for more detailed information):

```
<Files ~ "\.inc">
    Order allow,deny
    Deny from all
</Files>
```

- Set the desired maximum file upload size: `upload_max_filesize = 10M`

4.3.2 Installation guide - MySQL Database

The first step towards the proper functioning of the database is to download the latest release of MySQL from <http://www.mysql.com>. Most of the features used by the WEBDOC project are already supported by the version 3.23.34a, however in order to provide maximum stability the newest stable version is recommended. It is important to stress at this point that the WEBDOC project uses the InnoDB tables in place of standard MyISAM tables. Hence it is recommended that it is ensured that the InnoDB support is running. In order to perform a complete database installation the following steps are to be carried out:

- Download and install MySQL
- Create a user named "php" with the password of "php"²⁷
- Create a user that can access the database remotely from a host of your choice – this will allow a remote administration of the database from any PC using the MyCC program
- Start the database engine
- Create a database named WEBDOCV4
- Use the file `create_database_v4.sql` to create all the tables – it is enough to load it into the SQL editor and execute everything
- Use the file `inser_users_v4.sql` to insert some meaningful data into the database

These steps conclude the installation process. The server should be now online and working.

²⁷ User name and the password can be changed in the code of the PHP file `config.php`

5. CHAPTER FIVE – CONCLUSIONS AND THE FUTURE

5.1 Possible future development

Among many possible future system enhancements the first one to be mentioned are the ones that were not included into the system due to the lack of time. They were however foreseen and the appropriate framework has already been conducted. It means that the appropriate database tables were created, however they are not used at the moment by the PHP code. Among such functionalities are:

- File search by keywords functionality (table `keywords`)
- Copy/move file/folder functionality
- Email notification (table `email_not`)

The above mentioned functions were not implemented into the system due to the lack of time that could be spent on developing the project. Some other ideas have emerged whilst the development however they could not be even introduced partially as it happened with the ones mentioned. These ideas provide additional functionality to the server making it more flexible and an easier tool for the end-user. Such ideas were:

- The ability to copy files in the server's virtual file system
- The ability to move files in the server's virtual file system

There is also one more unfinished issue. It concerns the client side browser. Despite the attempts to make the interface as universal as possible it has not been possible to make it work seamlessly under the Netscape and Opera browsers. The increasing number of vain attempts to achieve the desired effect has led to conclusion that it might be useful to introduce a sort of browser-safe interface to the project. With help of this interface it would be possible to browse the project in any browser available. This issues became especially visible at the FH Aalen, where in one of the laboratories there were only computers equipped with the Netscape WWW browser.

Another idea concerned the database and the file system. Namely it is possible to store files directly in the MySQL database. This approach eliminates the need to introduce special functions for handling the physical files on server. Its main advantage is the fact that the system becomes more coherent and does not have any direct impact on the actual Operating System's file system. This is a severe security improvement. Moreover it is possible for anyone who has access to the server (as root) to view all the system files – this could be a possible security hole since it is not intended that the administrator would be able to read and modify all the documents stored on server. These modifications could easily lead to the database inconsistency, since the state it would represent would not reflect the actual state of the documents on the server.

This approach eliminates such danger. Now only the database administrator would have access to all the documents which is intentional. This change would surely have impact on the database performance since it would need to handle more requests and its size would be significantly larger.

With 10.000 documents each at the size of 1 MB we get a database table that is 10GB large. It is also important to stress that the documents would not be accessible for the server administrator, hence in case of database failure all data would be lost. Therefore it is very important to stress at this point the need for regular backup.

Yet it has to be bared in mind that storing file data in the database's tables makes several functions much easier to implement. The file copy function or file move function, for example, would be a question of just one update query in the database. In the present situation it involves also file system related PHP routines which might lead to possible security flaws.

To sum up - this idea is definitively worth taking a closer look at.

5.2 Conclusion

WEBDOC project proved to be successful implementation of web based document retrieval system. There are several factors that speak for the aforementioned thesis. First o all system has been implemented in two locations – one is the FH Aalen in Aalen, Germany. Second one is the State Psychiatric Hospital in Rybnik, Poland. Moreover, there are preparations to implement the system at the FH Ulm in Ulm, Germany. These successful realisations prove that customer is satisfied with the system.

There are several stress points which guarantee system's flawless functioning and customer satisfaction. First one of them is the reduction of redundant paperwork in the administration and everyday office use of the WEBDOC system. Second one is the continuous updating of the already working systems. Thanks to the well thought design and Object Oriented Approach, along with the modular construction of the system itself, it is extremely easy to develop new features and provide customer with new versions.

So far there were very few minor problems with the system, which thanks to the features mentioned before were very fast eliminated. It has to be stressed that there are almost no system drawbacks. Possible enhancements discussed in the previous chapter add new functionality to the system, however the already existing version meets all the customer requirements and is described as "*positively influencing the document interchange*" in the business units it has been implemented in.

WEBDOC system though not actually being a full implementation of Hospital Information System is a very good and one may say integral part of it. It does not implement specific administration functions or any patient management routines and yet by allowing undisrupted circulation of electronic documents and data it greatly reduces the need for paper work. Moreover it saves a lot of money by reducing the administration costs that would be normally spent on the document processing.

6. REFERENCES

- MySQL AB (2003). *MySQL Reference Manual for version 4.1.0-alpha*. Retrieved January 22, 2003, <http://www.mysql.com/documentation/mysql/bychapter/index.html>
- Apache HTTP Server Documentation Project (2003). *Apache HTTP Server Version 1.3 Documentation*. Retrieved January 22, 2003, <http://httpd.apache.org/docs/>
- Bakken, S S. et al. (18-04-2003). *PHP Manual*. Retrieved May 1, 2003, <http://www.php.net/manual/en/>
- Coar, K. *Adding PHP to Apache on Linux*, Retrieved December 12, 2002, <http://www.linuxplanet.com/linuxplanet/tutorials/1374/1/>
- Lim, J. (09-09-2000). *Apache on Windows*. Retrieved December 12, 2002, http://php.weblogs.com/apache_windows
- Google (2003). *Google Web Search Engine*, <http://www.google.com/>
- The PostgreSQL Global Development Group (2002). *PostgreSQL 7.3 Documentation*. Retrieved December 12, 2002, <http://www.postgresql.com/docs/view.php?version=7.3&idoc=0&file=index.html>
- SAP AG (2003), *SAP DB Library*, Retrieved December 12, 2002, <http://www.sapdb.org/7.4/htmhelp/e2/55683ab81fd846e10000000a11402f/frameset.htm>
- Innobase Oy Inc (2002), *InnoDB Homepage*, Retrieved December 12, 2002, <http://www.innodb.com/>
- Barker, R. (1990), *CASE*Method, Entity Relationship Modeling*, Addison-Wesley Pub. Co.
- Jelsoft Enterprises Limited (2002), *Web Design Forums.net*, <http://www.webdesignforums.net/>
- SuSE Linux AG (2002), *SuSE Support-Datenbank*, Retrieved February 12, 2003, <http://sdb.suse.de/de/sdb/html/>

- Robins, N. and Roberts, S. (September 1996). *Rethinking Paper Consumption*, Retrieved March 01, 2003, <http://www.iied.org/smg/pubs/rethink.html>
- Organization for Economic Co-Operation and Development (14 November 1996). *Report of the OECD workshop "Rethinking Paper Consumption"*, Retrieved March 01, 2003, <http://www.iied.org/pdf/gd97111.pdf>
- Economagic.com, (2003). *PPI: Pulp, paper, and prod., ex. bldg. paper; SA*, Retrieved March 05, 2003, <http://www.economagic.com/em-cgi/data.exe/blswp/wps091>
- Marchal, B. (1999). *Electronic Data Intechange on the Internet*, Retrieved March 24, 2003, http://developer.netscape.com/viewsource/marchal_edata.htm
- A.Rainio, P.Ahonen. (1994). *Workshop on EDI and Joint Use of Geographic Information - part B*, Retrieved march 24, 2003, <http://www.nls.fi/ptk/standardisation/2.html>
- World Wide Web Consortium (6 October 2000). *Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation 6 October 2000*, Retrieved March 05, 2003, <http://www.w3.org/TR/REC-xml>
- World Wide Web Consortium (15 October 2002). *Extensible Markup Language (XML) 1.1 W3C Candidate Recommendation 15 October 2002*, Retrieved March 05, 2003, <http://www.w3.org/TR/xml11/>
- Chung, E. and Dardailler , D. (April 9, 1997). *White Paper : Joint Electronic Payment Initiative (JEPI)*, Retrieved March 06, 2003, <http://www.w3.org/ECommerce/white-paper>
- UN Economic Commission for Europe. *United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport*, Retrieved March 03, 2003, <http://www.unece.org/trade/untdid/welcome.htm>
- The Accredited Standards Committee (ASC) X12, (2002). *UN/EDIFACT – The International EDI Standard*, Retrieved March 03, 2003, <http://www.x12.org/x12org/international/index.cfm>
- EDIFICE - Standardized Electronic Commerce forum (16 June 1999). *Internet EDI Implementation Guideline - ISSUE 1*, Retrieved March 03, 2003, <http://www.edifice.org/REP2/int1.pdf>
- Navarro, A. (2001), *Effective Web Design, Second Edition*, Sybex

- Garrand, T. (2001), *Writing for Multimedia and the Web, Second Edition*, Focal Press
- Flanders, V. (2002), *Son of Web Pages That Suck: Learn Good Design by Looking at Bad Design*, Sybex
- Wroblewski, L. (2002), *Site-Seeing: A Visual Approach to Web Usability*, Hungry Minds
- Armoni, A. (2002), *Effective Healthcare Information Systems*, Idea Group Publishing
- Rada, R. (2002), *Information Systems for Health Care Enterprises*, HIPAA-IT.com
- Mahmood, M, A. (2002), *Advanced Topics in End User Computing*, Idea Group Publishing
- Chen, Q. (2001), *Human Computer Interaction: Issues and Challenges*, Idea Group Publishing
- Khosrowpour, M. (2000), *Challenges of Information Technology Management in the 21st Century: 2000 Information Resources Management Association International Conference*, Idea Group Publishing
- German Research Association (DFG) (1998), *Annual Report 1997*, DFG, Bonn,.
(Deutsche Forschungsgemeinschaft (DFG): *Jahresbericht 1997*)

7 ATTACHEMENTS

This master thesis contains three attachments:

- CD-ROM containing the WEBDOC system. Apart from the WEBDOC system the CD-ROM contains an electronic version of this master thesis along with the presentation
- An Opinion provided by the CEO of the State's Psychiatric Hospital in Rybnik, Poland
- Summary in Polish