NTNU
Innovation and Creativity

Going Open
Building the Platform to Reach Out

Per Kristian Schanke

Master of Science in Computer Science
Submission date: July 2007
Supervisor: Reidar Conradi, IDI
Co-supervisor: Carl-Fredrik Sørensen, IDI
Audun Jensvoll, Keymind Computing AS

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

The project will build up a portal for releasing projects as Open Source. It will focus on the necessary choices and the criteria for getting a good portal.

Assignment given: 20. January 2007
Supervisor: Reidar Conradi, IDI

Going Open

Building the Platform to Reach Out

Per Kristian Schanke

Supervisor:
Carl-Fredrik Sørensen & Reidar Conradi

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

Abstract

This report presents the results of the development of a portal for open source software. The work is done in collaboration with Keymind Computing AS in context of the European ITEA project COSI[1].

The purpose of this project is to develop a portal so that companies that got commodity software they want to go open source with can do so without loosing control of the development. The portal is built up using already existing tools to fulfill as many tasks as possible.

The thesis also try to explain why making a portal for the release of open source by looking at the history of open source. Some of the focus here is on the development of the *Open Source 2.0* which i identified by the growing interest among software companies to release their software under and open license.

Keywords: Open Source, Open Source project portal

[1] Co-development using inner & Open source in Software Intensive products. Website: `http://itea-cosi.org`

Preface

This report is a masters thesis as the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU). The project is performed in context of the *Co-development using inner and Open source in Software Intensive products* (COSI) project which is an ITEA project.

I would like to thank Audun Jensvoll from Keymind Computing AS for valuable information about their needs for a portal.

I would also like to thank my supervisor Carl-Fredrik Sørensen for valuable input on the process, research methods, and information.

I would finally thank my fellow students Tor Erik Eide, for help with the design and the fall depth project, and Idar Borlaug, for help with technical problems regarding Linux.

Trondheim, 2007-07-30

Per Kristian Schanke

Contents

Abstract **III**

Preface **V**

Contents **VII**

List of Figures **XI**

List of Tables **XIII**

List of Code **XV**

I Context **1**

1 Introduction **3**

1.1 Motivation 3

1.2 Project Context 3

1.3 Report Outline 3

2 History **5**

2.1 Computers 5

2.2 College Days 5

2.3 GNU/Stallman 6

2.4 *nix 6

2.5 OSS 2.0 6

3 Research Design **7**

3.1 Literature Survey 7

3.2 Action research 7

3.3 Research goals 8

II Development of the portal **9**

4 Requirements **11**

4.1 Guidelines . . . 11

4.2 Rationale . . . 11

5 Design of the portal **13**

5.1 Design . . . 13

5.2 Language . . . 13

5.3 Data storage . . . 14

5.4 Auxiliary . . . 14

5.5 Extensions . . . 15

5.6 Main files . . . 15

6 Choices **17**

6.1 Technical skills . . . 17

6.2 Choices . . . 18

7 Implementation of the portal **21**

7.1 Setting up the system . . . 21

7.2 Develop the framework . . . 21

8 Problems during development **25**

8.1 Linux . . . 25

8.2 Mailserver on testplatform . . . 25

8.3 Mailman on Dreamhost . . . 26

9 Expected time to develop a portal **27**

9.1 Infrastructure . . . 27

9.2 Framework . . . 27

9.3 Tools . . . 28

9.4 Total . . . 28

III Conclusion **29**

10 Evaluation **31**

10.1 Research Question 1 . . . 31

10.2 Research Question 2 . . . 31

10.3 Research Question 3 . . . 31

11 Conclusion **33**

11.1 Future work 33

IV Appendix **35**

A Glossary **37**

B Bibliography **39**

C Index **41**

D Choices **43**

D.1 Linux 43

D.2 Versioncontrol 44

D.3 HTTP server 44

D.4 Mailinglist 45

D.5 Bugtracker 46

D.6 Forum 47

E Code **49**

E.1 Auxiliary - basic.inc.php 49

E.2 Language - main.inc.php 54

E.3 Language - news.inc.php 54

E.4 Data storage - news.inc.php 55

E.5 Main files - news.php 55

E.6 Extensions - mantis.inc.php 56

E.7 Design - top.inc.php 58

E.8 Main files - documentation.php 60

E.9 Configuration - config.inc.php 60

List of Figures

3.1 Cyclical Action Research model, [Davison et al. 2004] 7
7.1 The look and feel of the design 22
7.2 The design split into separate files 23

List of Tables

D.1 Comparison of certain Linux properties in different distributions 44
D.2 Active sites counted by Netcraft, June/July 2007 45

List of Code

5.1 General layout of *basic.inc.php* 15
5.2 Extensions example 15
5.3 Main file example 16
E.1 Auxiliary - basic.inc.php 49
E.2 Language - main.inc.php 54
E.3 Language - news.inc.php 54
E.4 Data storage - news.inc.php 55
E.5 Main files - news.php 55
E.6 Extensions - mantis.inc.php 56
E.7 Design - top.inc.php 58
E.8 Main files - documentation.php 60
E.9 Configuration - config.inc.php 60

Part I
Context

Chapter 1
Introduction

1.1 Motivation

Open source software has gained a large momentum in the recent years, with more and more commercial actors adopting open source as an important strategy. Several of these have released software as open source, or are considering such a move. When a commercial company release software as open source, they might want to keep more control of the project than the average hobby programmer. The objective of this project is to create a portal that enables companies to release their sourcecode under an open source license, but still control all the tools to continue development of the code. This portal will be based on a own developed framework, providing login and some simple features, and already existing tools for the more complex tasks, such as bugtracking and forums.

I will also present the history of open source and some of the companies working within the open source community. This research will be a literature survey.

1.2 Project Context

This project is written as a masters thesis at the Norwegian University of Technology(NTNU), Trondheim. This project is an obligatory part of the Masters Degree, and provides 30 ECTS credits, which responds to 100% of the work load in the semester.

The project is performed in the context of the COSI project, which is an ITEA project aimed at making the techniques and processes of open source known in companies developing software. The project is performed in cooperation with Keymind who are releasing their Keywatch software as open source.

1.3 Report Outline

Part I contains a description of the research and reasearch methods, and background infromation about open source. In Chapter 1, the research is defined. In Chapter 2, the history of open source is discussed. In Chapter 3, the research methods used in this project and the research goals are presented.

Part II describes the work done in this thesis. In Chapter 4, the requirements of the project is discussed. In Chapter 5, the overlaying design of the portal is presented. In Chapter 6, the process of creating the portal is presented, with an overview of expected use of resources. In Chapter 7, the steps taken to develop the portal is discussed. In Chapter 8, the major problems that occured during the implementation is highlighted. In Chapter 9, an overview of the time and resources needed to develop a similar portal is discussed.

Part III presents the conclusions that can be drawn from this thesis. In Chapter 10, an evaluation of the

work done is presented. In Chapter 11, the conlusion of the thesis is drawn, and a suggestion for further research is presented.

Part IV contains the appendixes of the report. Appendix A contains a glossary of words and phrases that are used in the thesis. Appendix B contains the bibliography. Appendix C contains an index of many of the most essential words. Appendix D presents the choices made during the development. Appendix E contains some of the essential parts of code.

Chapter 2
History

2.1 Computers

The history of computers is relatively short. And the history of computers is even shorter. As late as in 1945 there where no store-program computers[Patterson and Hennessy 1995; pp 1]. With the lack of storage space, there were also a no need for things like operating systems as every program in itself was responsible for everyhting it did. There was no other processes to fight our resources with.

With the first harddrive in 1956, the IBM 305 RAMAC[1], the possibilities of computers improved. With harddrives came more advanced programs; and the need to be able to start up and stop programs led to the development of the first operating systems. But most of these early computers were enormous in size, and not something anyone had access to.

One of the most important changes in the short history of computers came in 1969 when Ken Thompson found a PDP-7 to develop and play his game "Space Travel" on[Bach 1990]. As development of programs for the PDP-7 needed another computer to compile, Thompson and his friend Dennis Ritchie started to develop an operating system for the PDP-7 so that it would become self supporting. This eventually developed into Unix. The most important factor of the spread of Unix was the fact that AT&T at the time could not market computer products because of a decree they had signed with the US government. But this did not stop them from giving it out to universities who wanted it for educational purposes. By 1977 there were as many as 500 known installations of Unix, whereof 125 in universities. In 1977, Unix was moved onto another platform than the PDP it had been linked to until then.

2.2 College Days

At the colleges of the United States of America, the hacker culture started early. According to [Raymond 2001a] can the first hacker culture initiate at MIT in 1961. These early hackers defined what we today see the remains of in the open source communities.

With the advent of ARPAnet in 1969, the students and staffs of many of the leading American universities became able to communicate directly to each other through the computer. This opened unknown opportunities, and all the opportunities were grabbed. With ARPAnet came also a standardization body named RFC (Request For Comments)[2]. RFC is not a normal standardization body. There are no formal procedures that have to be followed. This informality made it possible for any person with some knowledge of an area of telecommunication and networking to come up with suggestions to solutions on known problems. And with the RFC's as a guide, anybody could build programs that communicated through the network. As long as both sides of a communication implemented the same RFC's, they could be developed in entirely different communities and still be expected to work.

[1] `http://en.wikipedia.org/wiki/IBM_305_RAMAC`

[2] *Request for Comments* - `http://en.wikipedia.org/wiki/Request_for_Comments` - Publisher: *Wikipedia* - Last accessed: *2007-07-30*

2.2.1 Unix

The timing of Unix was impeccable. Arriving at the scene at the same time as ARPAnet meant that it could be spread wide without to much work. Versions ended up everywhere. Up until 1984, when AT&T could take command over Unix again, all kinds of improvements and changes was made to it.

Among the additions to Unix were the *X Window System*. That was developed at MIT. According to [Raymond 2001a] was the success due to the fact that they spread their source freely,

2.2.2 BSD

With Unix back into the hands of AT&T, the Berkley Unix got its real shot at the headlights. The Berkley Unix had built-in support for ARPAnet protocols. This meant that it was on the leading edge in the rivalry between AT&T's Unix and the Berkley Unix.

2.3 GNU/Stallman

Richard M. Stallman (RMS) has become the one of the best known persons within the open source community the last two-three decades. With his organization, *Free Software Foundation* (FSF), he has developed many of the tools that currently see widespread use. RMS has pushed many of the projects that we today see the fruits of. RMS started the development of Emacs and GCC, and also created his own licence, the *GPL*. He also initiated the GNU project which had as it purpose to duplicate the Unix library and kernel so that any code developed for Unix also could run on his GNU operating system.

2.4 *nix

RMS was not alone when it came to the ideal of creating a free Unix clone. According to [Raymond 2001b] did the followers of Berkeley Unix do the same; only with the BSD license instead. But neither RMS or the BSD followers succeeded. To successfully create a free Unix clone, you needed a finn.

2.4.1 Linus

Linus Torvalds was the man needed to get the kernel to work. Linus used the libraries provided by the GNU project and built his own kernel. The name of this kernel would eventually be Linux. According to [Raymond 2001d] was one of the most important things Linus did was to release early, and often.

2.5 OSS 2.0

On 1998-01-22 did Netscape announce that they would release the source of their Netscape browser. This resulted eventually in the Mozilla browser, who later became Firefox. This was the first of many companies that later have released their software as open source. Some examples of this new wave of open source are Sun with its release of OpenOffice and coming release of Java, and IBM with the release of Eclipse,

Chapter 3
Research Design

3.1 Literature Survey

Conducting a literature survey is a good way to get a better understanding of the field that is being researched. As this research based on the prestudy from the fall semester 2006, [Eide and Schanke 2006], I have a good general knowledge of the open source community. Most of the literature survey will be based on the material from the fall, but it will be extended in the areas necessary.

3.2 Action research

Most of the work in this thesis will be action research, [Davison et al. 2004]. Action research is a formalized way of learning by doing. There are five steps to the action research:

1. **Diagnosis** Finding out as much as possible about the problem and find out what to do
2. **Action Planning** Find a solution to the problem and plan how to intervene
3. **Intervention** Go through with the plans
4. **Evaluation** Find out if the plans had the desired effect
5. **Reflection** Consider the effects of the iteration, and see if another iteration is needed

A presentation of the iterative process in action research can be seen in Figure 3.1.

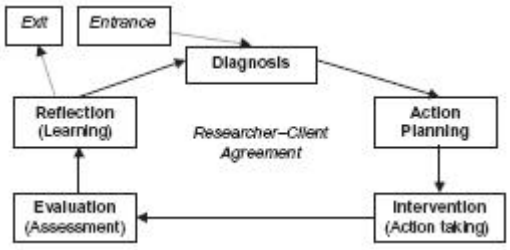


Figure 3.1: Cyclical Action Research model, [Davison et al. 2004]

In this project there will be three iterations:

1. Development of framework
2. Integration of first extension

3. Integration of seconf extension

3.3 Research goals

The main objective of this thesis is to:

> Develop a portal for the release of source code as open source. The portal shall be developed using tools available under an open license. Necessary procedures for accepting new users, creating new projects and administer them is also necessary.

R1 Describe and choose the necessary tools

R2 Develop a system of user handling

R3 Develop a system to handle extensions to the system

Part II
Development of the portal

Chapter 4
Requirements

This project never had any formal requirements regarding the portal. However, there were a few guidelines that were agreed upon by all the participants. This chapter will list up these guidelines and consider the rationale behind them.

4.1 Guidelines

R1 Possibility to run multiple projects with sub-projects

R2 Base the development on existing components where possible

R3 Develop the needed framework to enable:

- R3.1 easy user registration
- R3.2 simultanous login
- R3.3 easy project creation
- R3.4 easy component inclusion

R4 Use components released under an open license

4.2 Rationale

4.2.1 R1

The possibility to run multiple projects and sub-projects on the portal is based on the needs of Keymind to develop plugins. Instead of setting up multiple installations of the portal and having to make the users register on several places they get a single spot to relate to. But instead of creating new projects for different plugins, it should be possible to create sub-projects for the different plugins. This makes the connection between a project and a plugin easier to see and administer.

4.2.2 R2

Developing a portal containing different tools is a resouce intensive job; it is also aginst the reasons to develop this project. One of the major reasons for initializing this project is to ease the release of software that is considered comodity software under an open license. Most of the tools that is relevant to include in this project is can easily be considered comodity software. Considering the resource limitations in this project, it would be impossible to consider developing all the tools from scratch.

4.2.3 R3
Even though the project should be based on already existing projects will it always be a need for knitting everything together. This is mainly concerned with userhandeling and projecthandeling. But it also include the need for a framework for inclusion of tools.

4.2.4 R4
As the portal is made to release and promote open source tools, it is only logical to use open source tools in the development. As there might be need to modify or duplicate code from the different tools, it is only possible to do this when the original source code is released under an open license.

Chapter 5
Design of the portal

The design in this project is connected with the framework connecting everything together. In general can the project be said to be divided into six distinct parts:

- Design
- Language
- Data storage
- Auxiliary
- Extensions
- Main files

I will in this chapter describe the reasoning behind the division of the files and what they contain.

5.1 Design

The design files are supposed to contain the design elements for all the specific elements. This is to make the look-and-feel of the project easy to customize. Every separate design belongs in a separate folder in the *design* folder. The design folder also contain a main file *main.inc.php* that contain the common functions for the design files.

There are two mandatory files for any new design: *top.inc.php* and *bottom.inc.php*. These two files contain the information of all information about the common parts of the design.

The separate sub pages may have separate design files that describe the design of the elements that the page consist of. Examples of elements that may be put in these design files are elements like news posts, headers for news, login box, and other similar elements.

5.2 Language

The language files are supposed to make translation of the portal easy. No texts in the portal should be locked into a specific language. Every separate language belongs in a separate folder in the *language* folder. There is also a main file *main.inc.php* in the language folder. This file contains the common functions for language files; more specific the function *message* that takes one mandatory and one optional parameter. A more detailed description of this file can be seen in Appendix E.2.

Every language needs a separate file for the translation of every sub page. In addition is there a common file for texts that are common to all pages, such as error messages and names of sub pages.

5.3 Data storage

Data storage files are all files that are concerned with the manipulation of the contents of the page. Only one form of data storage will initially be implemented, MySQL, but the design of the portal is meant to be general so that switching to any other form of data storage is possible.

The division of the data storage is similar to that of the other elements; every sub page have a separate file. In addition, there is a common file *main.inc.php* that contain the common functions for the specific form of data storage. This includes functions such as *connect* and *checkaccess*.

5.4 Auxiliary

Auxiliary files include all files that contain common functions. The most important of these files is *basic.inc.php*. This file handles most of the necessary functions. It loads all configurations, extensions, and other files a sub page needs. It also provides a range of functions that is used on most sub pages. It also handles session data and, with that, user information.

```
<?php
/* Include the following files:
 *  -Configuration files
 *     - config.inc.php
 *     - local.config.inc.php (if exists)
 *  -Database files
 *     - tools/database/$CONFIG["db_type"]/main.inc.php
 *     - tools/database/$CONFIG["db_type"]/$module_name.inc.php (if exists)
 *  -Language files
 *     - language/$CONFIG["language"]/main.inc.php
 *     - language/$CONFIG["language"]/$module_name.inc.php
 */

//Connect to the database
connect();

//Initialize everything
init();

/* Load the extensions.
 * The used extensions are available in the variable
 * $EXTENSIONS from the configuration files.
 * Every extension has an controller
 * ($EXTENSIONS[$i]["controller"]) that resides in the
 * extensions folder that has to be loaded first.
 */
for ($i = 0; $i < count ($EXTENSIONS); $i++)
    require_once ($CONFIG["extension_path"]."/".$EXTENSIONS[$i]["↘
        →controller"]."inc.php");

/* basic.inc.php defines these functions:
 *   - displaypage ()
 *       Has the responsibility of showing the page.
 *   - check_logedin ()
 *       Checks if the user is loged in
 *   - init ()
 *       Initializes the variables.
 *   - createlink ($linkaddress, $parameters = false, $inprojectid = -1)
 *       Creates a link to another subpage ($linkaddress).
```

```
 *    - treeposition ()
 *       A tree representing the hierarchical position of the current
 *       project
 *    - login ()
 *       Loads the loginbox if not loged in, else information about the
 *       loged in person
 *    - validateemail ($mailaddress)
 *       Validates that $mailaddress is a valid emailaddress
 *    - notempty ($text, $minimum = 4, $maximum = 0)
 *       Checks if $text is not empty, and between the
 *       size requirements (if $minimum > $maximum, is not maximum used).
 *    - register_function ($hook_function, $custom_function)
 *       Used by extensions to hook their customized functions to
 *       the regular functions
 */
?>
```

Code 5.1: General layout of *basic.inc.php*

5.5 Extensions

Extensions are files that enable externally developed tools. Examples of this include *Bugtracker* and *Forum*. Extensions contain the necessary functions to log in, log out, user creation, project creation, and other tasks available in the main system. The basic setup of an extension can be seen in Code 5.5.

```
<?php
//Include the configuration file (if any)
require_once ("extensions/extensionname_config.inc.php");

//Register the functions that shall be called by the different hooks
register_function ("login", "login_extensionname");
register_function ("logout", "logout_extensionname");
register_function ("newuser", "newuser_extensionname");
register_function ("newproject", "newproject_extensionname");

//Implement the functions that was registered
?>
```

Code 5.2: Extensions example

The way the extensions are meant to work is that when a sub page perform a certain function, such as *login*, will the registered function of all extensions that have registered their own login function also be called. The consequence of this is that it is easy to handle many different forms of tools as long as their inner workings can be manipulated.

5.6 Main files

The main files are the pages that is responsible for the actual showing of a page. These files sets up all the content of the sub pages, checks for permissions, and stores data. Some examples of Main files are: News, download, and login. An example of a main file can be seen in Code 5.6

```
<?php
//The name of the sub page
$modulename = "modulename";
/* Load the basic files.
 * This automatically loads configurationfiles,
 * languagefiles, datastoragefiles and designfiles
 * based on the $modulename.
 */
require_once ("tool/basic.inc.php");

//Necessary commands to create the page

/* Call to a common function defined in basic.inc.php
 * that displays the page.
 * Everything in the $pagecontent variable will be vissible
 * in the main part of the page, while the content of the
 * $rightcontent will be shown in the righthand bar of the page
 */
displaypage ();

//Necessary functions
?>
```

Code 5.3: Main file example

Chapter 6
Choices

The implementation of this project has been roughly been divided into three separate parts:

1. Setting up infrastructure (operating system, web servers, etc)
2. Develop the framework
3. Modify the tools

I will now present my skills in these areas, then show what I did, and lastly go through the major problems that came up.

6.1 Technical skills

When evaluating the outcome of a certain project is it important to have a knowledge of all the factors that have an effect on the result. As I am going to set up, program and run this portal more or less on my own, knowledge of my skills, and level of expertise.

Linux

- Experienced with using desktop applications
- Experienced with normal use of terminal
- Some experience with installation of Linux

Apache

- Some experience setting up
- Some experience administration

SVN

- Some experience setting up
- Some experience using
- No experience connecting to Apache

PHP

- Some experience setting up
- Some experience connecting to Apache
- Experienced in using

MySQL

- Some experience setting up
- Some experience connecting to PHP
- Experienced in using

6.2 Choices

6.2.1 Infrastructure

As the project is supposed to make a webportal, there was a few essential choices to make at the start of the project:

1. Decide whether to:
   (a) set up a webserver
      - Choose hardware
      - Choose operating system
      - Choose webserver
      - Choose mailserver

   (b) use a webhotel
      - Choose webhotel
2. Choose programing language
3. Choose database type

Choosing programing language and database were the easiest questions. Among the programing languages there were basically four choices available:

1. PHP
2. Perl
3. Python
4. JSP

Based on my personal preference I ended up with PHP. Another reason to choose PHP is that many of the available open source tools are programmed in PHP, making the connecting of the different tools easier.

The choice of database was even easier than choice of programing language as there is one de facto database used in the open source world: MySQL. Other possible choices include PostgreSQL.

The remaining question on the list is by far the hardest as there are major advantages to both approaches. Setting up a separate webserver gives complete control of everything related to the portal, but it is harder to maintain. Using a webhotel makes the maintenance of irrelevant things - such as webservers, operating systems, etc - easier, but you get locked into certain tools. As the portal has to work on as many different platforms as possible, I have chosen to set up one development server where I control everything, and use a webserver to run a production version for Keymind.

Webserver
Setting up a webserver gives maximum control of your server. It is also by far the most work with regard to setting up and maintaining.

The hardware used in this project is a *ACER Aspire E380* with a *AMD Athlon 64 X2 4200+*[1].

The next choice to make is operating system. In this project I ended up with Ubuntu Linux as this is one of the easier Linux editions. And it is also one that I have some prior knowledge. For more reasoning behind the choice of Ubuntu, see Appendix D.1.

The choice of HTTP server was a pretty easy one. *Apache* is the de facto HTTP server on Linux platforms. Apache was installed via *Aptitude* in Ubuntu. The package I installed was the *LAMP* package that automatically installs *Apache2*, *MySQL* and *PHP*. More information about the choice of Apache can be found in Appendix D.3.

The hardest choice to make concerning the choice of software was mail server as this is an area that I am not accustomed to. The only thing I was quite certain about when it came to mail, was that *Mailman* would be my mailinglist handler. The choice of Mailman as mailinglist handler is due to its de facto status among the open source community. For more reasoning about the choice of Mailman, see Appendix D.4.

Even with the choice of mailinglist out of the way, the choice of mail server is hard. The software I ended up with was *Exim* due to the fact that Exim and Mailman supposedly[2] is meant to easily work together through correct configuration.

Webhotel
The requirements for a webhotel running the portal is actually quite strict. To be able to run all the tools that is chosen, it has to have support for the following:

- PHP - To run the portal itself
- Python - To run Mailman
- SVN - To run version control
- Mail server - To be able to send mail
- Shell access - To ease installation of packages and set up of access rights to files
- MySQL - As the backbone of many tools and the portal itself
- Unlimited, of close to unlimited, bandwidth
- 100Mb+ storage
- Reasonably priced

It is not many webhotels that actually offer all this in one package. On the top of our shortlist, we found Dreamhost to be the most practical choice. In addition to fulfilling all the requirements, Dreamhost also included Mailman in an easy configuration.

6.2.2 Tools
The choice of tools were based on different criterions. A more detailed reasoning behind all of the choices can be seen in Appendix D. The choices have tried to take into consideration what are the state of the art within the different tool categories.

The choice of mailinglist is pretty easy. Through the years, Mailman has become the more or less de facto software for mailinglists. The only disadvantage conserned with the use of Mailman is that it is programmed in Python, making it harder to connect with the framework and other tools in the system.

[1] Complete features: *CPU:* AMD Athlon 64 X2 4200+; *RAM:* 2GB DDR II; *HDD:* 2x250GB; *Chipset:* nForce 430; *Network:* 2xGigabit Ethernet integrated; *Graphics:* ATI Radeon X1650SE HM 512MB;

[2] *GNU Mailman* - `http://en.wikipedia.org/wiki/GNU_Mailman` - Publisher: *Wikipedia* - Last accessed: *2007-07-19*

Choosing versioncontrol comes down to two feasible alternatives: CVS and Subversion. As Subversion is thought of as a replacement for CVS, it become the natural choice.

The choice of bugtracker was a decision between going with the well known main contender, Bugzilla, or try to find a new alternative. As the choices of mailinglist and versioncontrol was pretty traditional, a more unknown contender became the bugtracker of choice; . The main reason for choosing Mantis was that it supported the necessary features and was implemented in PHP, making it easier to integrate.

The final choice made within the timeframe of this project was forum. The chosen forum was phpBB due to it beeing open sourced and had all the relevant features.

Chapter 7
Implementation of the portal

The development in this project has been divided into two distinct items: Setting up the system and tools; and programming the framework. I will go through the setup and development of these two different types of problems separately.

7.1 Setting up the system

Setting up the infrastructure is pretty straight forward with Ubuntu. Ubuntu use the package manager Aptitude that handle dependencies between packages. Most of the installation is to choose the correct packages.

The major packages installed and configured where:

- LAMP (Linux, Apache, MySQL, PHP)
- Exim
- Mailman
- Subversion

In addition was Tomcat and Java installed to satisfy the needs of [Aaslund and Larsen 2007] who are running a somewhat similar project.

Mailman and Exim was configured to work together, as was Subversion and Apache.

7.2 Develop the framework

The development of the framework can be divided into six different parts: Design, language, auxiliary, data storage, extensions, and main files. The initial development used a simple design and few essential common functions. And later on the functionality became more and more advanced.

7.2.1 Iteration 1 - Show it

Getting the visual part of the site seemed to be a natural starting point for a first iteration. The first file to be implemented was *basic.inc.php*. This contained a basic shell that loaded a design and placed the content were it should be in the design.

The second step was to get this showing of a page into a context. The initial context was the initial frontpage of the portal. This file loaded *basic.inc.php* and used the functions and variables already existing here.

Next up was enabling a language module. First was a generic function named *message* created to pick up the correct string and manipulate it. The code for this can be seen in Appendix E.2. Next up was loading this file and the running modules language file. The task of this was given to *basic.inc.php* that took the name provided by the module and the configured language from the language file, and loaded the correct language file. With this addition, changes to the names of any part of the messages given by the system is controlled by a configuration variable.

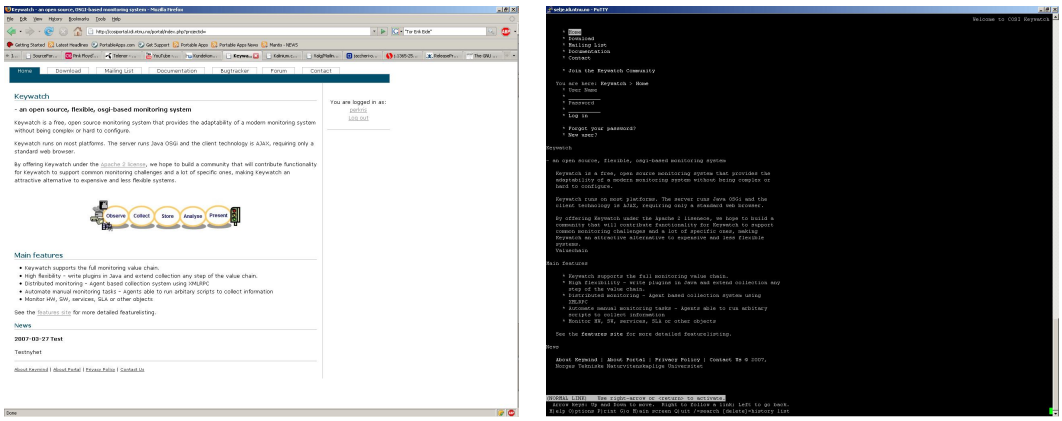
Finally in the first iteration, connecting to the data source. In the initial installation, this is a MySQL database. The database connection for a sub page is based on two files: One generic file that handle connecting and disconnecting, and a database module for the sub page that perform the necessary operations.

With this iteration we get a system that shows everything you need and can be used as a stepping stone for development of the other sub pages.

7.2.2 Iteration 2 - User handling

When a working shell is up and running, it is time to allow the user to log in. The first step in this direction is to enable create a login sub page that can handle this. All the user information is stored in the database. The password is encrypted with the *PASSWORD* function in MySQL. This ensure that the password is safe even if a sucessfull break in. Upon login will a SQL call compare the encrypted password in the database with the user supplied password by encrypting the user supplied password with the same function. This makes for a pretty good password consistency and safety. No user or administrator can ever read out the password in cleartext from the database.

With the login functioning, the next step is to see to that the user has access rights. The access rights of a certain user using a certain feature at certain project may be determined in many different ways. The first way is by the users own access rights on that feature on that project. The second way is by the users access rights on the portal itself. The third way is by the general access rights on that feature on that project. Finding a good way to store this without using to much space depends on how it is optimized. It is one thing that simplifies this work, and that is treating project 1 as the portal project. This way it is only needed to keep track of two tables: user rights to projects, and default project rights.



(a) In Firefox (b) In Lynx

Figure 7.1: The look and feel of the design

After fidling with access rights, the new design is integrated. The design is developed by [Eide 2007]. The look and feel of the design can be seen in Figure 7.1. After some splitting, the design is broken into the pieces that can be seen in Figure 7.2.

7.2.3 Iteration 3 - Adding the extensions

With user handling in place, extension handling should be handled. The first thing on the menu is to identify what steps have to be made to include an extension into the portal so that the user can login on many systems at the same time. The first step to this is to identify the key functions. The four

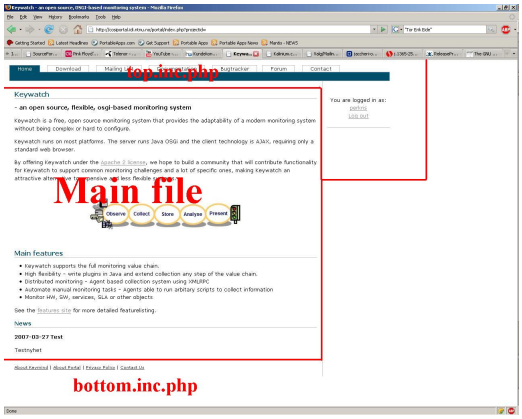


Figure 7.2: The design split into separate files

major ones are: login, logout, create user, and create project. The consequence of this is that to all these existing functions in the system there have to be able to run hooks from external entities that handle the functionality for the tools. This is done by creating functions that can be called dynamically. These functions are implemented in an controller for every tool.

The controllers contain all the functions that are needed by the system to perform the essential tasks. These functions are registed in an array that again can be used to call the functions dynamically when needed. It is important that no two function have identical names as this crash the system.

Chapter 8
Problems during development

As with all software projects, there were many problems arising. Some of them were major, but most were easy to work around. I will go through some of the problems.

8.1 Linux

Due to the fact that the hardware of the test computer was close to the bleeding edge at the time of installation of Linux, much of the hardware was not supported in the Ubuntu version we initially installed (6.10). The main problem was the ethernet card[1] that was not available. The first attempt to fix this problem was to download the drivers from the providers web pages and compile them as a module to the kernel. Even after installing the driver, the network interface would not work. After some consultation with friends and web pages, it seemed to be a problem with the Linux kernel running. The 2.6.17 did not support the ethernet card, but the 2.6.19 had that support.

Installing a new kernel fixed all the hardware problems. The only problem connected to the installation of the new kernel was that it needed to be compiled. The compilation followed a description on HowtoForge[2]. According to this description, the compilation should go smooth if choosing to use the configuration file of the already running kernel. This was not the case. Even though the already existing kernel had support for the SATA harddrives in the system, these were not included in the configuration file. So, when trying to load Linux with the new kernel was impossible. After finding and fixing this problem, everything went smooth with the Linux install.

8.2 Mailserver on testplatform

Installation of Exim mailserver on the testplatform was pretty straight forward when using Aptitude. Configuration was not as easy. None of the mails sent through the system got through to their target. After configuration and tweaking of the webserver some of the mails got through. All mails sent to a server on the local subnet[3] got through, but any mail to external accounts were blocked. After some further configuration changes the reason became obvious. The NTNU policy on outgoing mail is that they have to go through the schools spam filters. As sending mail out of the school network was not essential on the testserver, no further work was made to fix this.

[1] Marvell Yukon Chipset based Ethernet Controller

[2] *How To Compile A Kernel - The Ubuntu Way* - `http://howtoforge.com/kernel_compilation_ubuntu_p2` - Publisher: *HowtoForge* - Last accessed: *2007-07-30*

[3] With and address of *user@*.ntnu.no*

8.3 Mailman on Dreamhost

After setting up the portal on Dreamhost, a mailinglist was also set up. This was set up through the Discussion List interface that is on the administration pages of the Dreamhost account. After some testing everything seemed fine and the portal was set in production. The first problem came with the first mail from an external user. His first mail ended up in the archive, but never went out to the registered members of the mailinglist. Checking this mail showed that it was a normal mail sent with base-64 encoding. After getting the mail from the archive, one of the registered users answered the mail. This mail went to all the users on the mailinglist. Going through the setup of the mailinglist showed no configuration errors on our side, so we contacted Dreamhost support. Dreamhost could not find any problems on their side either.

I suspect that the problem is conected with the way the mail was sent. After some direct communication with the external user, I saw something strange in the mails I got from him. All of his mails contained only a MIME part with the base64 encoded text. Normally do mails contain a part with the text in clear format and one encoded part if they contain some encoded material. As this was not the case with these mails may Mailman have stripped off the MIME part of the mail and tried to send it on. And as the mail without its MIME part would be an empty mail, Mailman, or the underlaying mailserver, refused to send it. The reason for Mailman to strip the MIME part is unknown as the configuration of Mailman told it to never strip anything of the mail. Dreamhost could not see any faults in their setup either, so this error proved impossible to find the source of.

Chapter 9
Expected time to develop a portal

The time needed to implement a portal depends on different factors. I will in this section set up some estimates on time needed to get a working portal up and running. All the estimates consider starting from scratch and a user with average knowledge of the fields considered.

9.1 Infrastructure

Setting up the necessary infrastructure depends on the choices made on operating system and software. Setting up Linux from scratch with an Apache webserver, Exim mailserver, MySQL, and SVN may take anything from a day to a couple weeks depending on everything from package management to hardware support.

To ensure that everything works, a minimum of one day should be used to set up each. The reason for this is that most of the programs needs extensive configuration and testing to make sure that they are set up correctly and safe. One week is what should be considered a minimum for setting up everything. If any of the hardware components are new on the market, additional time should be added to cope with new drivers and modules that might need to be compiled in and tested.

9.2 Framework

The framework is divided into three different areas: Useradministration, sub pages, and design. Of these are useradministration the most technically difficult. The useradministration do not only need to handle the login and logout, but also keep track of access rights across different tools, projects, and sub pages. Designing and implementing such a scheme can easily take a full week depending on the complexity of the structure and the generality wanted.

Designing the web page itself is probably the most important thing. No matter how well made your features are, if the design is bad, the user will probably never return. Making the design both look good and easy to use takes a lot of time. Coding the HTML and CSS from the ground up without relying on already existing design is easily a weeks work. The difficulty of breaking the design into parts that can be handled in the code is relatively small. Making a generic way to handle the code is slightly harder, but if a template system is defined before starting the design, both the design and coding can work towards a common interface, making the breaking easier, and making it easier to change the design later on without modifying everything.

The complexity defines the time it takes to make a sub page. An easy sub page, such as News, may only take a day to make, while more advanced, such as a documentation tool, may take anything from three days to weeks. In a project such as this, we should only implement those sub pages that is easier to make than import. Anything with an expected development time of more than a week is probably easier to find a tool that already handles and use this instead.

9.3 Tools

Adding a tool to a portal should be made as easy as possible. The way it is designed in this project is pretty straight forward, and do not require major changes to the source of the tool itself. Changes should be made externally, making updates of the tools easier to integrate.

Even though integration of new tools is made as simple as possible, they take time. There are especially four factors that play in: login system, creation of new users, creation of new projects, and design. To integrate a new tool, the developer should have intimate knowledge of the system he integrates. Getting this may easily take a the better part of a week. Most of the tools used in this project has been based on PHP and , making integration as easy as possible. Most tools today use a combination of cookies and sessions to keep track of users. The easiest way to integrate the tools are to create own functions that set these variables as needed.

Even though the tool itself has all the functions needed and most of the integration consists of copying these, is it not easy to integrate a new tool. All functions will need customization; and proving that they work as well as original is not as easy as it might seem.

The part that usually is the hardest is to integrate the design of the portal to a tool. Most of the tools that are relevant is usually used as standalone packages. They often has advanced template systems themselves, and you need insight into the system to integrate your own design with them.

All in all will the integration of a tool to a portal take a minimum of two weeks. This might seem like a misuse of resources as the tools often are aged pretty fast. The advantage of the approach is that few of the tools incorporate major design changes for a specific function within a major release – and often not between major releases – so when there is a new release, normally only an update of the tool and a short testing period is needed to keep in touch with the state of the art.

9.4 Total

The complete setup of a portal on your own webserver will take in the range of twelve to fifteen weeks with a reasonable amount of tools. The sub pages and tools included in this calculation is: News, download, documentation, mailinglist, bugtracker, and forum. One or two weeks can be shaved off the infrastructure part when using a webhotel.

Part III
Conclusion

Chapter 10
Evaluation

This chapter contains the evaluation of the portal compared with the research goals from Chapter 3:

RQ1 Describe and choose the necessary tools

RQ2 Develop a system of user handling

RQ3 Develop a system to handle extensions to the system

10.1 Research Question 1

> Describe and choose the necessary tools.

During this research, I have considered different tools that are necessary for an open source portal. As we described in our fall depth-study, [Eide and Schanke 2006], there are certain tools that have to be included in a portal. These are: Mailinglist, bugtracker, forum, and documentational tools. Tools for news and download are also needed, but these are going to be programmed from scratch as their functionality is pretty simple.

Out of the four tools mentioned, I have performed an analyses to find the best mailinglist, bugtracker, and forum. As expected from the start were there not time to include an advanced documentational tool, so a minimalistic system is implemented instead.

10.2 Research Question 2

> Develop a system of user handling.

Designing a user handling system that works well is pretty tricky. The system that I have developed in this project is a decent basis for a user handling system, but it is not optimal. Its strengths is in its simplicity. It is in principle easy to add on new extension and get the system to handle the control of access rights. The problem is that the graphical interface to do this is not implemented yet. The support is in the database, and to a certain degree in the code, but it lacks in the interface. Until this is improved will the system be impractical.

10.3 Research Question 3

> Develop a system to handle extensions to the system.

This is the only system that at the moment works as it is supposed to. All tools can be installed as they normally are installed. As soon as they are installed, the user install the controller for the tool and the tool can be installed as an extension. During the installation will all projects and users be copied to the tool. The only problem is that user rights will have to be set automatically too. The logical way to do this would be to give project owners administrative rights to their project on the tool, and everybody else a defaul, low access. But this is something that can be configured in the install routine for the specific tool.

All extensions are loaded automatically during the running of any script, so any maintenance work that is needed can be scheduled in at load time.

Chapter 11
Conclusion

This project has shown that it is possible to create a portal for releasing open source software using already existing tools for handling many of the more advanced features. The development of such a portal requires extensive planing of the administration of projects and users as these are the main features of the framework that need to be built in the bottom.

The advantages of using already existing tools is immediately obvious. Creating many of the features needed on a portal, such as a forum and bugtracker, is to complicated to do without experienced developers. At the same time is the technology behind such tools considered commodity software. There is no need to start the development of such features when they already exist freely available.

The portal developed as part of this project has shown some of the strengths and weaknesses of developing a portal instead of using an existing alternative. As all the contents of the portal in itself is comodity software, such a portal should be made available for further development. The major problem with making the portal as generic as possible is that it gets hard to find a generic way to administer users. All the relevant tools have their own independent administrative interfaces. The access rights of the individual tools should be configured once, and then later when modifying the access of a single user should be made able from a single user administration page instead of having to go into tool administration. Getting this setup and working is harder than it might seem.

11.1 Future work

Future work with this portal is in the region of extensions and user administration. As it is now, the user administration is not intuitive. Some work is needed to make this easy to use. There is also a possible need for more extensions. The first obvious extension to add is some kind of documentation handling tool. A possible type of tool is a wiki.

Part IV
Appendix

Appendix A
Glossary

***nix** Operating systems that follow the same design priciples as *Unix*. The most widespread is *Linux*, but different versions of *BSD*, *Minix*, and *Solaris* is also considered to be **nix*.

Apache An open source http-server.

Aptitude Package manager for Ubuntu Linux.

Bleeding edge An item that has just come out on the market.

BSD Operating system based on the Unix distribution developed at the University of California, Berkeley, in the 1970s. There is currently many differnt forks of the BSD code, with *FreeBSD* as the probably most widely deployed. *Mac OS X* is partly based partly on FreeBSD.

Cookie An identifier sent between a web server and browser to keep track of the value of a variable.

COSI Co-development using inner and Open source in Software Intensive products. Project run by ITEA to make the techniques and processes of open source known in companies developing software.

CSS Cascading Style Sheets. Used to describe the presentation of the data written in a markup language.

Dreamhost Large providor of webhotels.

Exim Open source mail server.

FSF Free Software Foundation - Organization started by Richard Stallman to promote open source software.

Hacker Individual with the ability to create and modify programs.

HTML HyperText Markup Language. Markup language used in webpages.

ITEA The International Test and Evaluation Association. Educational organization founded to further the exchange of technical information in the field of test and evaluation. (`http://www.itea.org/`)

JSP *JavaServer Pages* - Script language used for dynamic webpages. Based on Java

LAMP Linux, Apache, MySQL and PHP. This is one of the most used setups for web servers.

Linux A range of operating systems based on the kernel developed by *Linus Torvalds*. The development started in 1991 and it is today one of the best known open source projects.

Open License A license that allows anyone to read and modify the sourcecode of a program. The best known licenses are: GNU General Public License (GPL), Apache, Mozilla Public License (MPL), and BSD.

Open Source A way to develop software were anyone can read and modify the sourcecode of a program. Programs released under an open license is considered to be open source software.

Operating system Set of computer programs that manage hardware and software resources in a computer.

PDP-7 Early computer

PHP *PHP: Hypertext Preprocessor* - Script language used for dynamic webpages.

Proprietary Software that is released under a license that do not allow other than the company developing the software to view and modify the code.

Shell access Full access to a systems commandline interface.

Session A way to keep track of the state of the communication between a server and a client.

Windows Operating system develped by Microsoft. Proprietary.

Appendix B
Bibliography

Aaslund, K. and Larsen, S. (2007). OTS-Wiki: A Web Community for Fostering Evaluation and Selection of Off-The-Shelf Software Components. Master's thesis, Norwegian University of Science and Technology.

Bach, M. H. (1990). *The Design of the UNIX Operating System*. Prentice-Hall PTR.

Collins-Sussman, B., Fitzpatrick, B. W., and Pilato, C. M. (2004). *Version Control with Subversion*. O'Reilly.

Davison, R. M., Martinsons, M. G., and Kock, N. (2004). Principles of Canonical Action Research. *Information System Journal*, 14(1):65–86.

Eide, T. E. (2007). Study of the Release Process of Open Source Software. Master's thesis, Norwegian University of Science and Technology.

Eide, T. E. and Schanke, P. K. (2006). Going Open: Guidelines for Commercial Actors to Release Software as Open Source. Master thesis prestudy.

Fogel, K. (2005). *Producing Open Source Software: How to Run a Successful Free Software Project*. O'Reilly.

Patterson, D. A. and Hennessy, J. L. (1995). *Computer Arcitecture; A Quantative Approach*. Morgan Kaufmann Publishers.

Raymond, E. S. (2001a). *A Brief History of Hackerdom*, chapter 1. In [Raymond 2001c].

Raymond, E. S. (2001b). *Homesteading the Noosphere*, chapter 4. In [Raymond 2001c].

Raymond, E. S. (2001c). *The Cathedral & the Bazaar - Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly.

Raymond, E. S. (2001d). *The Cathedral and the Bazaar*, chapter 2. In [Raymond 2001c].

Appendix C
Index

*nix, *37*, 43

ACER, 19
AMD, 19, 44
Apache, 17, 19, 21, 27, *37*, 44
APT, *see* Aptitude
Aptitude, 19, 21, 25, *37*, 44
ARPAnet, 5, 6
AT&T, 5, 6

Bleeding edge, 25, *37*
BSD, *37*
  FreeBSD, 44
  History, *see* History, BSD

Cookies, 28, *37*
COSI, V, 3, *37*
CSS, 27, *37*

Dreamhost, 26, *37*, 43

Exim, 19, 21, 25, 27, *37*, 45

Files
  Auxiliary
    basic.inc.php, **14–15**, 21, 22, 49–54
  basic.inc.php, 14
  Configuration
    config.inc.php, 60–61
  Data storage
    main.inc.php, 14
    news.inc.php, 55
  Design
    bottom.inc.php, 13
    main.inc.php, 13
    top.inc.php, 13, 58–60
  Extensions
    example, **15**
    mantis.inc.php, 56–58
  Language
    main.inc.php, 13, 54
    news.inc.php, 54–55
  Main files
    documentation.php, 60
    example, **16**
    news.php, 55–56
Free Software Foundation, 6, *37*, 49
FSF, *see* Free Software Foundation, *see* Free Software Foundation

Hacker, *37*
History
  BSD, 6
  College Days, 5–6
  Computers, 5
HTML, 27, *37*

Infrastructure, 18–19
  Webhotel, 19
    Shell access, 19, *38*
  Webserver, 19
ITEA, III, V, 3, *37*

JSP, *37*

Keymind, III, 3, 11, 18
Keywatch, 3

LAMP, 19, *37*
Linux, V, 17, 27, *37*
  choosing, 43–44
  Debian, 44
  kernel, 25
  problems, 25
  Ubuntu, 19, 21, 37, 44
  Utuntu, 25

MIT, 5, 6
MySQL, 14, 17–19, 21, 22, 27, 28, 46, 47, 49

NTNU, V, 3, 25

Open License, III, 8, 11, 12, *37*, 47
  Apache, 37
  BSD, 6, 37
  GPL, 6, 37, 47, 49
  MPL, 37, 46
Open Source, *37*
Operating system, 19, *38*
  Linux, *see* Linux

PDP-7, 5, *38*
PHP, 17–19, 21, 28, *38*, 47
Proprietary, *38*
Python, 19

Research goals, 8
RFC, 5
Ritchie, Dennis, 5
RMS, *see* Stallman, Richard M.

Session, 28, *38*
Solaris
    OpenSolaris, 44
Stallman, Richard, 37
Stallman, Richard M., 6
Subversion, *see* Tools → Versioncontrol → SVN
SVN, *see* Tools → Versioncontrol → SVN

Thompson, Ken, 5
Tools
    Bugtracker, 15, 46–47
        Bugzilla, 20, 46–47
        Mantis, 20, 47
    Forum, 15, 47–48
        Invision Power Board, 47
        MVNForum, 47
        phpBB, 20, 48
        vBulletin, 48
    Mailinglist, 19
        Mailman, 19, 21, 25–26, 45–46
        MajorDomo, 45–46
    Versioncontrol
        CVS, 20, 44
        SVN, 17, 19–21, 27, 44
Torvalds, Linus, 6, 37

Unix, 5, 6
User
    Administration, 27

Windows, *38*
    why not, 43

X Window System, 6

Appendix D
Choices

This appendix contain the arguments for choosing different pieces of software. They will contain a small introduction to the criterias set to ground and the actual choices.

D.1 Linux

When choosing an operating system for an open source project is a *nix version the logical choice. The reason for this is that by choosing a non-free operating system, you end up on a collision course with the principles of open source.

D.1.1 Why not Windows

The reasons for not even considering Windows as the operating system for the portal are:

1. It breakes with the philosophy of open source development to offer an open system from a proprietary platform,
2. Most webhotel-providers use Linux as operating system on their servers,
3. Most of the software considered are originally made for *nix before it is ported to Windows, thus making the *nix version more thoroughly tested.

There are no reasons for not using Windows if that is your only practical option. All tools and software suggested here should run on Windows, or have an alternative implemented for Windows.

D.1.2 Criteria

For this project, we have two different test environments. The first is a server at the school that we ave full access to. The other is a webhotel at *Dreamhost*. Dreamhost uses *Debian*[1] as operating system on their servers.

The server I have full access to is an Acer Aspire e380. This is going to be used as both a testserver for new releases, and as a production server for the portal. The server is the basic found on Acers homepages[2] , but with 4Gb of RAM.

D.1.3 Selection

Considering the criterias listed in the previous section is it hard to make a choice as all the systems are based on much of the same philosophy and technology. As I need to make a choice, I will start by choosing an operating system based on the Linux kernel. This excludes OpenSolaris and FreeBSD. This is mainly

[1] `http://wiki.dreamhost.com/index.php/Linux` (Last accessed 2007-01-24)

[2] `http://www.acer.no/acereuro/page4.do?sp=page117\&dau22.oid=20687\&UserCtxParam=0\&GroupCtxParam=0\&dctx1=12\&CountryISOCtxParam=NO\&LanguageISOCtxParam=no\&ctx3=94\&ctx4=Norge\&crc=1508460289#inu57_38428` (Last accessed 2007-01-23)

Property	Minimum demand	Debian	OpenSolaris	Ubuntu	FreeBSD
Processor type	AMD Athlon64 Dual Core	Yes	Yes	Yes	Yes
Package handeling	Easy	APT	pkgadd	APT	Packages/Ports
Security updates	Frequent	Frequent	Frequent	Frequent	Frequent
Provided packages	Stable	Yes	Yes	Yes	Yes
Cost	Free	Free	Free	Free	Free

Table D.1: Comparison of certain Linux properties in different distributions

due to my knowledge of the systems, not of any technological reasons. To choose between the different Linux versions, I use the *Linux Distribution Chooser*[3] to make the final choice. When considering myself as an advanced used, not caring what type of installer, installing as a server on a shining new 64-bit desktop with a stableset of packages, the recomandation ends up with *Debian*, *Fedora*, *Mandriva*, and *Ubuntu*. Actually choosing is then up to the small differences. I choose *Ubuntu* due to the 18 month guaranteed security updates, the easy installation, and to test it on a slightly different platform from the one used by *Dreamhost* to check if everything works on different setups.

D.2 Versioncontrol

The choice of version control software comes down to the two most popular at this time, CVS and Subversion, [Fogel 2005; 53]. Choosing between them is as much a question about taste and preference, as of actual technical differences between the two tools.

D.2.1 Selection

Ignoring taste and preference, there are few distinct reasons to choose one of the tools over the other. Both tools are open source, and both feature most of the same features. According to [Collins-Sussman et al. 2004], Subversion was meant as a replacement of CVS. The technical advantages to Subversion over CVS is with regard to version handling of items such as directories and metadata. The fact that Subversion handle all types of files equal is also a big advantage. For these reasons are Subversion chosen ahead of CVS.

D.3 HTTP server

The choice a HTTP server depends on the use of the server. Our server has to be able to run server side script as PHP and also be able to display the content of a Subversion repository.

D.3.1 Criteria

The HTTP server used have to be able to handle web pages made with PHP, Perl, and Python. It should also be able to authenticate and connect to a Subversion repository. And it should be open source.

D.3.2 Selection

With the criteria that were set, it is only one feasible option for a web server: Apache. Apache is by far the most used according to Netcraft. Table D.2 show that there are 1.5 times as many active web sites using Apache than the closest competitor. As it also fulfills all the other requirements.

[3] `http://www.zegeniestudios.net/ldc/` (Tested 2007-01-24)

[4] *July 2007 Web Server Survey* - `http://news.netcraft.com/archives/2007/07/09/july_2007_web_server_survey.html` - Publisher: *Netcraft* - Last accessed: *2007-07-30*

Developer	June 2007	Percent	July 2007	Percent
Apache	29066283	51.11	29208937	49.98
Microsoft	19925417	35.03	20761521	35.53
Google	3749705	6.59	4321502	7.39
lighttpd	237195	0.42	247493	0.42
Sun	186362	0.33	229960	0.39
Zeus	218512	0.38	215429	0.37

Table D.2: Active sites counted by Netcraft[4], June/July 2007

D.4 Mailinglist

Mailinglists have a long following with regard to open source projects. You can not run an open source project without a mailinglist.

D.4.1 Criteria

The criteria for choosing a mailinglist is as follows:

- Easy to add new mailinglists
- Easy to moderate
- Stores a copy of all messages in an archive
- Possibility to limit access to list
- Use a database for administration (easier to administer from common framework)

D.4.2 Alternatives

Mailman
GNU Mailman is one of the best known mailinglists.

Advantages

- Wide spread use (easier to get support)
- Easy to sign up on
- Closely integrated with Exim
- May be used with any mailserver
- Easy to moderate

Disadvantages

- Developed in Python, with some C
- Sends out password to the users once a month by default
- Hard to integrate with rest of framework

MajorDomo
MajorDomo is another of the most used mailinglists. It lacks an official webinterface, but there are some non-official, such as MajorCool

Advantages

- Wide spread use (easier to get support)
- Webinterface (through MajorCool)
- Easy to moderate through mail

Disadvantages

- Developed in Python
- No official webinterface
- Hard to integrate with rest of framework

D.4.3 Selection
Considering the advantages and disadvantages of both the considered tools, the choice ends up on Mailman. The reason for this is that it is widely deployed and under constant development. It also has a webinterface that is easy to use.

D.5 Bugtracker

D.5.1 Criteria

- Easy to install and configure
- Possibility for external authentication
- Easy to update and follow the status of a bug
- Open source compatible license
- Implemented in PHP, Perl, Python or similar serverside language
- Flatfile of MySQL data storage

D.5.2 Alternatives

Bugzilla
Advantages

- Well known in open source community
- MySQL is a possible database choice
- Open licence (MPL)
- Offers support for external authentication
- Easy to add new projects

Disadvantages

- Not to easy to modify
- Hard to set up

Mantis
Advantages

- Open license (GPL)
- Implemented in PHP - easier to manipulate
- Clean roadmap
- Easy to follow the life of a bug
- Supports MySQL

Disadvantages

- Not as many, or as advanced, features as Bugzilla

D.5.3 Selection

Bugzilla is by far the best known bugtacker. Bugzilla also sets the standard for what needs to be included in open source bugtrackers. Mantis has the advantage of being the smaller tool. The reason to choose Mantis over Bugzilla is to try out an unknown tool, in addition to the fact that Bugzilla has almost become to generic for its own good.

D.6 Forum

Forums are a type of software that is obviously easy to create as there are so many different choices available. But I have to choose one.

D.6.1 Criteria

- Released under an open license
- Implemented in PHP (easier to modify)
- Possibility to create sub forums
- Possible to subscribe to threads
- Reasonable active development
- Uses MySQL for data storage

D.6.2 Alternatives

MVNForum

- Implemented in JSP
- Not developed since july 2006

Invision Power Board

- Not released under an open license

vBulletin

- Not released under an open license

phpBB

- Implemented in PHP
- Released under an open license
- Should be easy to create subforums
- Possible to subscribe to threads
- Still under active development (2.0.22 came in December 2006; 3.0 RC1 came in May 2007)
- Supports MySQL

D.6.3 Selection

Even though there are other forums available are phpBB a safe bet. It is one of the best known, and is used in many different situations.

Appendix E
Code

This appendix contain some code examples from the portal. It contain some of the essential files, such as *basic.inc.php* and the main language file *main.inc.php*. It also contain a set of files needed for the creation of a sub page, namely **News**. All the necessary files to present this is: *news.inc.php* (Language), *news.inc.php* (Data Storage - MySQL), and *news.php* (Main file). Also included is *documentation.php* that presents special type of main file, and *top.inc.php* which is one of the main design files, and *config.inc.php* which is the main configuration file.

All the code is released under the GPL 2 license[1]

E.1 Auxiliary - basic.inc.php

```
<?php
if (!isset ($module_name))
{
    $module_name = "Home";
}

require_once ("config.inc.php");
if (file_exists ("local.config.inc.php"))
{
    require_once ("local.config.inc.php");
}
/* Including the basics for database connection */
require_once ("tools/database/".$CONFIG["db_type"]."/main.inc.php");
if (isset ($module_name))
{
    require_once ("tools/database/".$CONFIG["db_type"]."/".$module_name."."↘
        →inc.php");
}
/* Including the basic language packages needed */
require_once ("language/".$CONFIG["language"]."/main.inc.php");
if (isset ($module_name))
{
    require_once ("language/".$CONFIG["language"]."/".$module_name."."inc.↘
        →php");
}

connect ();
init ();
```

[1] *GNU General Public License - Version 2* - `http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt` - Publisher: *Free Software Foundation* - Last accessed: *2007-07-30*

```
$hooks = array ();
$hooks["login"] = array ();
$hooks["logout"] = array ();
$hooks["newuser"] = array ();

for ($i = 0; $i < count ($EXTENSIONS); $i++)
{
    require_once ($CONFIG["extension_path"]."/".$EXTENSIONS[$i]["↘
       →controller"]."inc.php");
}


$projectid = !isset ($_GET["projectid"]) || !is_numeric($_GET["projectid"↘
   →]) ? $CONFIG["default_project"] : $_GET["projectid"];

if ($CONFIG["log"])
{
    require_once ("tools/log.inc.php");
}

$documents = ($projectid == 1 ? "portal" : "keywatch");

$logedin = check_logedin ();
$pagecontent = "";
$rightcontent = "";

if ($CONFIG["show_login"])
{
    $rightcontent .= login ();
}

function displaypage ()
{
    global $CONFIG, $pagecontent, $rightcontent, $EXTENSIONS;
    require_once ("design/".$CONFIG["template"]."/top.inc.php");
    if (is_array ($pagecontent))
    {
        foreach ($pagecontent as $line)
        {
            echo $line;
        }
    }
    else
    {
        echo $pagecontent;
    }
    require_once ("design/".$CONFIG["template"]."/bottom.inc.php");
}

function check_logedin ()
{
    //TODO: Check how long time since user loged in last time
    return $_SESSION["logedin"];
}

function init ()
```

```
{
    session_start ();
    session_register (userid);
    session_register (remember_me);
    session_register (session_active);
    session_register (returning);
    session_register (logedin);
    session_register (username);

    header ("Cache-Control: no-cache, must-revalidate");
    header ("Pragma: no-cache");
    header ("Expires: Thu,01 Jan 1970 00:00:00 GMT");

    if (!isset ($_SESSION["session_active"]))
    {
        $_SESSION["session_active"] = true;

        if (isset ($_COOKIE["returning"]))
        {
            $_SESSION["returning"] = $_COOKIE["returning"];
            getreturninginfo ($_SESSION["returning"]);
        }
        else
        {
            $_SESSION["userid"] = -1;
            $_SESSION["username"] = "";
            $_SESSION["returning"] = 0;
            $_SESSION["remember_me"] = false;
            $_SESSION["logedin"] = false;
        }
    }
}

function createlink ($mylink, $parameters = false, $inprojectid = -1)
{
    global $projectid;

    $linkprojectid = $projectid;

    if ($inprojectid != -1)
    {
        $linkprojectid = $inprojectid;
    }

    $mylink .= ".php?projectid=$linkprojectid";

    if (is_array($parameters))
    {
        $params = count ($parameters);
        for ($i = 0; $i < $params; $i++)
        {
            $name = $parameters[$i]["paramname"];
            $contents = $parameters[$i]["contents"];
            $mylink .= "&amp;$name=$contents";
        }
    }
```

```
    return $mylink;
}

function treeposition ()
{
    global $projectid, $module_name;

    $_r = array ();

    $parentsid = $projectid;
    $i = 0;

    $_r[$i]["type"] = $module_name;
    $_r[$i]["projectid"] = $projectid;
    $_r[$i]["name"] = message ($module_name);
    $i++;

    while ($parentsid != -1)
    {
        $parent = getparent ($parentsid);
        $parentsid = $parent["parentid"];
        $_r[$i]["type"] = "index";
        $_r[$i]["projectid"] = $parent["id"];
        $_r[$i]["name"] = $parent["name"];
        $i++;
    }

    return $_r;
}

function operation ($operation)
{
    return getoperationid ($operation);
}

function login ()
{
    if (!$_SESSION["logedin"])
    {
        $actionlink = createlink ("login");
        $newuserlink = createlink ("newuser");
        $forgotlink = createlink ("forgot");
        $_r = <<<LOGINFORM
            <form action="$actionlink" method="post" style="white-space:↘
                →nowrap;">
                <div class="login">
                    <ul>
                        <li>User Name</li>
                        <li><input type="text" name="username" size="10" ↘
                            →/></li>
                        <li>Password</li>
                        <li><input type="password" name="password" size="↘
                            →10" /></li>
                        <li>Remember Me: <input type="checkbox" name="↘
                            →remember" checked="checked" /></li>
                        <li><input name="submit" type="submit" value="Log ↘
                            →in" size="10" /></li>
```

```
                    </ul>
                    <ul>
                        <li><a href="$forgotlink">Forgot your password?</a>
                            →></li>
                        <li><a href="$newuserlink">New user?</a></li>
                    </ul>
                </div>
            </form>
LOGINFORM;
    }
    else
    {
        $profilelink = createlink ("profile");
        $username = $_SESSION["username"];
        $logedinas = message("logedinas");
        $_r = <<<PROFILE
            <div class="login">
                <ul>
                    <li>$logedinas: <a href="$profilelink">$username</a></
                        →li>
                    <li><a href="logout.php">Log out</a></li>
                </ul>
            </div>
PROFILE;
    }

    return $_r;
}

function validateemail ($mailaddress)
{
    return ereg ("^[^@ ]+@[^@ ]+\.[^@ \.]{1,4}$", $mailaddress, $trashed);
}

function notempty ($text,$minimum = 4, $maximum = 0)
{
    $limits = "$minimum,";
    if ($maximum > $minimum)
    {
        $limits .= "$maximum";
    }

    $_r = ereg ("^[a-zA-Z0-9 .,;:]{{$limits}}", $text);
    return $_r;
}

function register_function ($hook_function, $custom_function)
{
    global $hooks;

    $count = count ($hooks[$hook_function]);

    $hooks[$hook_function][$count] = $custom_function;
}

?>
```

E.2 Language - main.inc.php

```
<?php

/*
 * This is a function that sets up messages.
 * It takes 1 or 2 paramters. The first is the message name
 * and the optional second is alternative parameters that the
 * message may contain.
 */
function message ($message_name, $parameters = 0)
{
    global $MESSAGES, $module_name, $CONFIG, $DEBUG;
    /* Checks if it really is a possible message */
    $message = isset ($MESSAGES[$message_name]) ? $MESSAGES[$message_name]↘
        → : false;

    if ($message)
    {
        $i = 0;
        if ($parameters != 0)
        {
            for ($i = 0; $i < count ($parameters); $i++)
            {
                $paramnr = "@@".($i + 1);
                $message = str_replace ($paramnr, $parameters[$i], ↘
                    →$message);
            }
        }
    }
    else
    {
        $message = str_replace ("@@1", $CONFIG["admin_email"], $MESSAGES["↘
            →message_error"]);
        if ($CONFIG["debuglevel"] >= $DEBUG["med"])
        {
            $message .= str_replace ("@@1", $message_name, $MESSAGES["↘
                →message_error_extension"]);
            $message .= str_replace ("@@2", $module_name, $message);
        }
    }

    return $message;
}

?>
```

E.3 Language - news.inc.php

```
<?php
$MESSAGES["news_header"] = "News";
$MESSAGES["addnews"] = "Add news";
?>
```

E.4 Data storage - news.inc.php

```
<?php
function fetchingress ($projectid, $number, $start = 0)
{
        global $dbhandle;

    $_r = false;
    $result = mysql_query ("SELECT * FROM news WHERE ProjectID=$projectid ↘
        →AND Active=1 ORDER BY NewsID DESC LIMIT $number", $dbhandle);

    if (is_resource ($result))
    {
        $i = 0;
        while ($row = mysql_fetch_array ($result))
        {
            $_r[$i]["id"] = $row["NewsID"];
            $_r[$i]["header"] = $row["Header"];
            $_r[$i]["ingress"] = $row["Ingress"];
            $i++;
        }
    }

    return $_r;
}
?>
```

E.5 Main files - news.php

```
<?php
$module_name = "news";
require_once ("tools/basic.inc.php");

$pagecontent = getnews ();

displaypage ();

function getnews ()
{
    global $CONFIG, $projectid;
    $_r = "";

    $news = fetchingress ($projectid, 100);

    $_r = "<h2 class=\"header\">".message ("news_header")."</h2>";
    $_r .= "<div class=\"news\">";

    if (is_array ($news))
    {
        for ($i = 0;$i < count ($news); $i++)
        {
            $params = array ();
            $params[0]["paramname"] = "newsid";
            $params[0]["contents"]  = $news[$i]["id"];
            $link = createlink ("readnews", $params);
```

```
            $_r .= "<h2><a href=\"$link\">".$news[$i]["header"]."</a></h2>
                →";
            $_r .= "<p class=\"rowBody\">".$news[$i]["ingress"]."</p>";
        }
    }

    if (checkaccess ($projectid, $_SESSION["userid"], 1) == 1)
    {
        $link = createlink ("addnews");
        $_r .= "<p><a href=\"$link\">".message ("addnews")."</a></p>";
    }

    $_r .= "</div>";

    return $_r;
}
?>
```

E.6 Extensions - mantis.inc.php

```
<?php
require_once ("extensions/mantis_config.inc.php");

register_function ("login", "login_mantis");
register_function ("logout", "logout_mantis");
register_function ("newuser", "create_user_mantis");
register_function ("newproject", "newproject_mantis");

function login_mantis ($username)
{
    global $MANTIS_CONFIG;

    $mantis_dbh = mysql_connect ($MANTIS_CONFIG["db_host"], $MANTIS_CONFIG
        →["db_username"], $MANTIS_CONFIG["db_password"], true);

    mysql_select_db ($MANTIS_CONFIG["db_database"], $mantis_dbh);

    $result = mysql_query ("SELECT cookie_string FROM ".$MANTIS_CONFIG["
        →user_table"]." WHERE username='$username'", $mantis_dbh);

    $row = mysql_fetch_array ($result);

    $cookie_string = $row["cookie_string"];

    setcookie ($MANTIS_CONFIG["cookie_name"], $cookie_string, time () +
        →365 * 24 * 60 * 60);

    mysql_close ($mantis_dbh);
}

function logout_mantis ()
{
    global $MANTIS_CONFIG;

    setcookie ($MANTIS_CONFIG["cookie_name"]);
}
```

```
function create_user_mantis ($userinfo)
{
    global $MANTIS_CONFIG;

    $mantis_dbh = mysql_connect ($MANTIS_CONFIG["db_host"], $MANTIS_CONFIG↘
        →["db_username"], $MANTIS_CONFIG["db_password"], true);

    mysql_select_db ($MANTIS_CONFIG["db_database"], $mantis_dbh);

    $username     = $userinfo["username"];
    $password     = $userinfo["password"];
    $mail         = $userinfo["email"];
    $realname     = $userinfo["realname"];
    $enabled      = $MANTIS_CONFIG["enabled_default"];
    $access_level = $MANTIS_CONFIG["access_level_default"];

    $cookie_string = auth_generate_unique_cookie_string_mantis ($email.↘
        →$username, $mantis_dbh);

    $usertable = $MANTIS_CONFIG["user_table"];

    $result = mysql_query ("INSERT INTO $usertable
                                (username, email, password, date_created, ↘
                                    →last_visit,
                                 enabled, access_level, login_count, ↘
                                     →cookie_string, realname)
                            VALUES
                                ('$username', '$mail', '-', NOW(), NOW(),
                                 $enabled, $access_level, 0, '↘
                                     →$cookie_string', '$realname')", ↘
                                     →$mantis_dbh);

    mysql_close ($mantis_dbh);
}

function auth_generate_unique_cookie_string_mantis ($seed, $dbh)
{
    $cookie_string = "";
    do
    {
        $cookie_string = auth_generate_cookie_string_mantis ();
    } while (!auth_is_cookie_string_unique_mantis ($cookie_string, $dbh));

    return $cookie_string;
}

function auth_generate_cookie_string_mantis ()
{
    $tval = mt_rand (0, mt_getrandmax ()) + mt_rand (0, mt_getrandmax ());
    $tval = md5 ($tval).md5 ($tval);

    return substr ($tval, 0, 64);
}

function auth_is_cookie_string_unique_mantis ($cookie_string, $dbh)
{
```

```
    global $MANTIS_CONFIG;

    $usertable = $MANTIS_CONFIG["usertable"];

    $result = mysql_query ("SELECT COUNT(*) AS c FROM $usertable WHERE ↘
       →cookie_string='$cookie_string'", $dbh);

    $count = 0;

    if (is_resource ($result))
    {
        $row = mysql_fetch_field ($result);
        $count = $row["c"];
    }

    return ($count == 0);
}

function newproject_mantis ($projectname)
{
}

?>
```

E.7 Design - top.inc.php

```
<?php
function selected ($module)
{
    global $module_name;
    if (!Strcmp ($module, $module_name))
    {
        echo "class=\"selected\"";
    }
}

function whereami ()
{
    $_r = "";
    $breadcrumbs = treeposition ();

    if (is_array ($breadcrumbs))
    {
        $first = true;
        for ($i = count ($breadcrumbs) - 1; $i >= 0 ; $i--)
        {
            if (!$first)
            {
                $_r .= " &gt; ";
            }
            $link = createlink ($breadcrumbs[$i]["type"], false, ↘
               →$breadcrumbs[$i]["projectid"]);
            $_r .= "<a href=\"$link\">".$breadcrumbs[$i]["name"]."</a>";
            $first = false;
        }
    }
```

```
    return $_r;
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta name="keywords" content="keywatch, open source, systems monitoring,↘
   →oss, networking, software, cosi, keymind, osgi, network monitoring, ↘
   →monitoring" />
<meta name="description" content="An open source OSGI-based monitoring ↘
   →system"  />

<title>Keywatch - an open source, OSGI-based monitoring system</title>
<link rel="stylesheet" type = "text/css" href="design/keywatch/keymind.css↘
   →" />
</head>

<body>
<div id="container">
    <div id="top">
        <div id="header">
        </div>
        <div id="topNavigation">
            <ul>
                <li <?php selected ("index"); ?> title="Keywatch Home"><a ↘
                   →href="<?php echo createlink ("index"); ?>">Home</a></↘
                   →li>
                <li <?php selected ("download"); ?> title="Download ↘
                   →Keywatch Source Code and Binaries"><a href="<?php ↘
                   →echo createlink ("download"); ?>">Download</a></li>
                <li <?php selected ("mailinglist"); ?> title="Subscribe to↘
                   → the Keywatch Mailinglist and become a Part of the ↘
                   →Keywatch Community"><a href="<?php echo createlink ("↘
                   →mailinglist"); ?>">Mailing List</a></li>
                <li <?php selected ("documentation"); ?> title="Read more ↘
                   →about Keywatch and Get help here"><a href="<?php echo↘
                   → createlink ("documentation"); ?>">Documentation</a↘
                   →></li>
                <?php for ($i = 0; $i < count ($EXTENSIONS); $i++)
                        {
                                echo "<li><a href=\"".$EXTENSIONS[↘
                                   →$i]["address"]."\">".↘
                                   →$EXTENSIONS[$i]["name"]."</a↘
                                   →></li>";
                        }
                ?>
                <li <?php selected ("contact"); ?> title="Contact Keymind,↘
                   → Keywatch Developers and COSI-participants"><a href="↘
                   →<?php echo createlink ("contact"); ?>">Contact</a></↘
                   →li>
            </ul>
        </div>
        <div id="join">
            <!--<ul>
```

```
                <li><a href="newuser.php">Join the Keywatch Community</a>
                    →></li>
            </ul>-->
        </div>
        <!--<div id="breadcrumbs">
            You are here: <?php echo whereami (); ?>
        </div>-->
    </div>
    <!--End Top-->

        <!--Rightmenu-->
    <div id="right">
        <div id="rightMenu">
            <?php
                echo $rightcontent;
            ?>
        </div>
    </div>
    <!--End Rightmenu-->
<div class="boxin">
    <div id="center">
```

E.8 Main files - documentation.php

```
<?php
$module_name = "documentation";
require_once ("tools/basic.inc.php");

if (isset ($_GET["sub"]))
{
    $page = $_GET["sub"];
    if (file_exists ("documents/$documents/$page.inc.php"))
    {
        require_once ("documents/$documents/$page.inc.php");
    }
    else
    {
        require_once ("documents/$documents/documentation.inc.php");
    }
}
else
{
    require_once ("documents/$documents/documentation.inc.php");
}

$pagecontent = contents ();

displaypage ();
?>
```

E.9 Configuration - config.inc.php

```
<?php
$CONFIG["language"] = "en";
$CONFIG["template"] = "design_template";
```

```
$CONFIG["max_loginattempts"] = 0;
$CONFIG["loginattempts_waittime"] = 10;
$CONFIG["admin_email"] = "admin@do.main";
$CONFIG["portal_name"] = "Portal name";
$CONFIG["portal_address"] = "http://do.main/";
$CONFIG["send_mail_from"] = "From: ".$CONFIG["portal_name"]." <user@do.↘
    →main>";
$CONFIG["portal"] = array ("do.main", "altdo.main");

/* Database configuration */
require_once ("db_config.inc");

$CONFIG["nr_news_on_frontpage"] = 5;
$CONFIG["password_minlength"] = 6;

$CONFIG["portaladmin"]       = 10;
$CONFIG["portalmoderator"]   =  9;
$CONFIG["portalwriter"]      =  8;
$CONFIG["projectadmin"]      =  5;
$CONFIG["projectmoderator"]  =  4;
$CONFIG["projectwriter"]     =  3;
$CONFIG["projectmember"]     =  2;
$CONFIG["portalmember"]      =  1;
$CONFIG["guest"]             =  0;

/* Operations available */
$CONFIG["writeaccess"]         =  1;
$CONFIG["changeothersaccess"] =  2;
$CONFIG["readothersprofile"]  =  3;

$OPERATION["addnews"]  = 1;
$OPERATION["sendmail"] = 2;

$DEBUG["low"] = 1;
$DEBUG["med"] = 2;
$DEBUG["high"] = 3;

$CONFIG["debuglevel"] = $DEBUG["low"];

$CONFIG["log"] = true;
$CONFIG["show_login"] = true;

$CONFIG["extension_path"] = "extensions";

$i = 0;
$EXTENSIONS[$i]["controller"] = "mantis";
$EXTENSIONS[$i]["name"] = "Bugtracker";
$EXTENSIONS[$i]["address"] = "mantis/";
$i++;
?>
```