Espoo–Vantaa Institute of Technology
Degree Programme
in Digital Information Provision

Jani Tarvainen

Bug Tracking Tool for Browser-based Applications

Bachelor's Thesis 28 April 2004

Instructor: Sven Borgers, Technology Manager
Supervisor: Aarne Klemetti, Principal Lecturer
Language Advisor: Marianne Kiekara, Lecturer

ESPOON–VANTAAN TEKNILLINEN AMMATTIKORKEAKOULU	INSINÖÖRITYÖN TIIVISTELMÄ

Tekijä Otsikko Sivumäärä Aika	Jani Tarvainen Selainpohjaisten sovellusten virheiden seurantajärjestelmä 71 sivua 28.4.2004
Koulutusohjelma	Digital Information Provision
Ohjaaja Valvoja	teknologiapäällikkö Sven Borgers yliopettaja Aarne Klemetti

Insinöörityössä suunniteltiin ja toteutettiin virheiden seurantajärjestelmä selainpohjaisia sovelluksia varten. Tavoitteena oli luoda helppokäyttöinen järjestelmä, johon projektin jäsenet kirjaavat kehitettävässä järjestelmässä huomaamansa virheet. Marmalade 1.0 järjestelmässä virheet tallennetaan tietokantaan, jota kehitystyöstä vastaavat henkilöt hallinnoivat selainpohjaisen käyttöliittymän avulla. Hallinnointiliittymä mahdollistaa virheiden seuraamisen projekteittain. Yksittäisen virheen tilaa vaihdetaan työn etenemisen myötä. Mahdolliset tilat ovat avoin, varattu ja suljettu.

Virheiden kirjaus tapahtuu JavaScript-tekniikalla rakennetun raportointielementin avulla. Tämä elementti rakennettiin täysin riippumattomaksi seurattavan järjestelmän palvelinohjelmistosta, käytetystä ohjelmointikielestä sekä tietokannasta. Elementti liitetään seurattavaan järjestelmään HTML-sivupohjia muokkaamalla. Sivupohjiin lisätään script-elementti, joka lataa raportointielementintoiselta palvelimelta. Projekti- ja paikkatietojen kirjaus ja ylläpito tapahtuu URL-osoittimien avulla. Selain yhdistää seurattavan järjestelmän tuottaman HTML-datan ja lomake-elementin. Käyttäjälle prosessi näkyy seurattavan järjestelmän näkymään lisättynä raportointilomakkeena.

Työ tehtiin digitaalisia markkinointiviestinnän palveluita tuottavalle Morning Digital Design Oy:lle. Työssä perehdyttiin virheidenseurannan teoriaan sekä tietokantapohjaisten verkkosovelluksien suunnitteluun ja rakentamiseen. Erityisen haastavaa oli palvelin- ja selainpohjaisten skriptikielien yhdistäminen. Työ ei kata järjestelmän käyttöönottoa yrityksen kehitysprosessissa.

Hakusanat	Virheiden seurantajärjestelmä, bugi, PHP, MySQL, Apache, JavaScript, RSS, XML, LDAP, GPL, Bugzilla, ClearQuest

ESPOO–VANTAA INSTITUTE OF TECHNOLOGY

ABSTRACT

Author Name of Thesis Pages Date	Jani Tarvainen Bug Tracking Tool for Browser-based Applications 71 28 April 2004
Degree Programme	Digital Information Provision
Instructor Supervisor	Sven Borgers, Technology Manager Aarne Klemetti, Principal Lecturer

The goal of the thesis project was to plan and implement a bug tracking tool for browser-based applications. The aim was to create an intuitive system for project teams to report bugs noticed in the developed system. In the Marmalade 1.0 system, the errors are reported into a database, which is administered by the developers. The developers administer the database using a browser-based tool. The administration interface allows the bugs to be monitored by project. An individual bug's status is updated as work progresses. The possible statuses are open, reserved and closed.

The reporting of bugs is done using a reporting client built with JavaScript technology. The client was built to be independent of the server software, programming language and the database of the tracked system. The client is integrated to the system under development by modifying the HTML templates. A script-tag that loads the client is added to the templates. Project and location data is maintained by using URL pointers. The browser combines the HTML data and the JavaScript client. To the user the process is shown as a floating reporting-element in the view of the tracked system.

The project was done for Morning Digital Design Oy, a Finnish provider of digital marketing services. Knowledge of the theory of common bug tracking practices and methods were gained. The implementation was technically challenging and further experience of web application development was acquired. Especially the implementation of the floater by combining server-side and client-side scripting proved to be challenging. The project did not cover the introduction of the tool to the development process of the company.

Keywords	Bug tracking tool, bug, PHP, MySQL, Apache, RSS, XML, LDAP, GPL, Bugzilla, ClearQuest

Table of contents

1. Introduction

Software bugs are familiar to both the developers and users of software. Matthew A. Telles [1, p. 56] offers the following definition for a bug: “Bugs are behaviors of the system that the development team (developers, testers and project managers) and customers have agreed are undesirable.” Due to human errors and bad specifications, bugs will continue to be a part of software development projects.

Tracking bugs is important for several reasons. Business wise bugs are considered expensive since they damage the company’s reputation and delay projects: “In 1994, Capers Jones conducted a survey of approximately 4,000 software projects. The survey asked the participants for reasons for schedule overruns. One of the most common reasons reported was poor quality.” [1, p. 63]

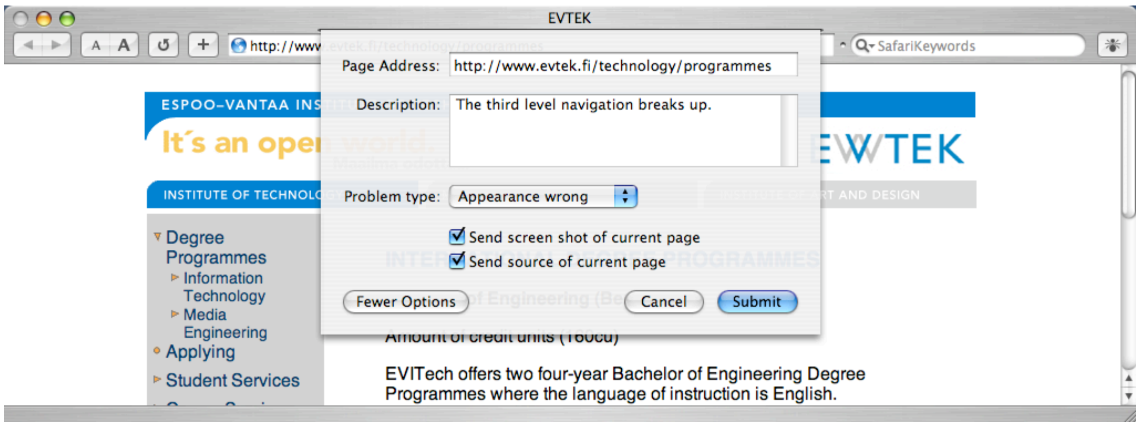
In addition, the survey also showed that many projects are simply cancelled due to an overwhelming amount of problems. From the human point of view, the people involved in projects with numerous problems get frustrated, demoralized and lose interest.

While web application development is a relatively new branch of software engineering, many of the traditional software engineering practices apply to it: “Following proper software engineering techniques (or modifying them for web development) can have a substantial impact on overall system usability primarily because it enables developers to achieve higher quality.” [2, p. 343]

The Marmalade 1.0 bug tracking system (also referred to as Marmalade) aims to help development teams to report and keep track of errors in projects. The system is meant for projects that utilize a browser-based interface such as a content management system or a web site. A browser-based system is, however, not a pre-requisite so it can be used to keep track of virtually any software project.

The system can be roughly divided into two components: reporting and administration. Found flaws are reported using the reporting interface and managed using the administration interface. The aim was to make the reporting of errors as convenient as possible. In addition, it was ideal to send as much automatic data as possible.

Apple Computer, Inc. has successfully applied a similar method in debugging their Safari web browser (public beta launched in January 2003). The application has an integrated bug reporting interface that allows users to report various problems. It has advanced features such as a built-in option to send a screen capture to demonstrate rendering errors. The sent error reports are logged to a database that the developers can access. The reporting interface is shown in Image 1.


Page Address: http://www.evtek.fi/technology/programmes

Description: The third level navigation breaks up.

Problem type: Appearance wrong

[✓] Send screen shot of current page

[✓] Send source of current page

Image 1 - Screen capture of the Safari browser's bug reporting interface

The client

The system was made for Morning Digital Design Oy (from now on referred to as Morning), a Finnish provider of digital marketing services. The work was done in September-December 2003 as an independent project.

The company has a relatively long history, dating back to 1988. Throughout the years the company has undergone multiple mergers and operating names. The current form was reached in January 2001 when To The Point Oy and TJ Group Branding and Advertising units were united. The shares of the formed company were distributed equally to the previous owners. Both Edita Oyj and TJ Group Oyj hold 50% of the shares.

With such a long history, the service offerings and products of the company have evolved through the years. As markets change the previous focus from producing CD-ROM multimedia products has switched to a broader direction: “Morning is a service company that focuses on consulting, planning and producing works for digital marketing communications” [3]

The company employs over 50 people in two offices in Helsinki and Tampere. The turnover for the year 2001 was EUR 4 million and EUR 4.5 million for the year 2002.

Motivation and objectives

Morning has multiple products that are either completely browser-based or have a browser-based administration interface. These include the Morning CMS system for content management on the web and the more versatile SmartStore CMS for publishing content for the following channels:

- Retail store terminals
- Mobile devices
- Internet
- CD-ROM / DVD

The main motivation for building the Marmalade system was the fact that there were no uniform means to keep track of the problems in the products being developed. Many project managers keep Microsoft Excel charts or similar tools to keep track of the problems. The requests for fixing the problems were given in person or by e-mail.

Introducing uniform reporting templates (examples shown in APPENDIX E) could make the process more fluent, but ultimately these methods of tracking bugs become a problem. For example, when a group of people is doing work on a project, time can be wasted by searching for problems that have already been addressed: “Were you able to find the reported bug? If not, remember that bugs don’t just go away; either they’re hiding, or they have been fixed already.” [4, p. 193]

The objective was to improve the workflow of the development process by providing a tool that would be used by the whole project team to keep track of problems and errors.

Structure of thesis

Chapter 2 contains a general description of the development process as a whole.

Chapter 3 describes the basic requirements set for the Marmalade system.

Chapter 4 elaborates the planning phase and briefly introduces the technologies used to build the system and the reasons they were chosen.

Chapter 5 explains the implementation in detail, including problems encountered.

Chapter 6 describes how to deploy the system to track a browser-based application.

Chapter 7 analyses the process in general and draws conclusions such as what should have been done differently and what could be done to improve the system.

APPENDIX A, B and C contain screen captures of the final application.

APPENDIX D contains a table of commercial bug tracking tool license fees.

APPENDIX E contains two examples of traditional bug reporting forms.

APPENDIX F contains the file structure of the final application.

2. The development process

The process had three major stages:

1. Setting the requirements
2. Planning
3. Implementation

The requirement stage defines what the product must be able to accomplish and what features it should have. Benchmarking similar existing systems is a useful method in setting the requirements. The requirements set at the beginning of the project are subject to minor changes during the planning and implementation stages.

In the planning stage, the methods to reach the requirements are chosen. This includes planning the interactions in the system and selecting the technical components:

- Programming language(s)
- Database
- Server platform

During the implementation stage, the specifications made in the planning stage will be used to create the system. When the application is considered to be in its final form or very near it, it is moved from the development platform to the production one.

The implementation phase also encloses the testing of the product as each added functionality is tested after it is built. Testing is an important part of the quality process, which should be continuous all the way through the project: "Quality assurance (QA) processes should occur throughout the production process. QA needs to start at the beginning of a project and never end." [1, p. 369]

Additional development phases may be launched after the product is released.

3. System requirements

The aim of the project was to create a system that would enable the users to log errors found in browser-based applications. This database of errors is administered using a web browser. To further specify the desired functionality some specific requirements were set. The requirements are shown in Table 1.

Table 1 - Requirements for the Marmalade system

Requirement	**Explanation**
Bug reporting must be convenient	A bug database is useful only if bugs are reported. By making the process as convenient as possible, the probability of reporting is increased.
The system must be independent of the server platform of the tracked application	The system must be able to track systems built on various server platforms or even static XHTML-pages.
The system must support multiple projects	The system must support multiple projects and allow a single user to work on several projects at the same time.
The system must support customized reports	Users must be able to create customized queries to the database using the following options: • Project • Bug status • Bug category • Bug priority • Keyword

Requirement	**Explanation**
The reporting client should log additional data automatically	The reporting client should automatically send as much automatic data as possible to make the reproduction of errors as simple as possible: “Without duplicating the environment of the problem, you cannot be sure that you have fixed the problem.” [1, p. 289]
The system must support attachments	Attachments allow the reporter to describe the reported error with files such as: • Screen captures • Content documents • Corrected images
The system must log bug category and status	The system must keep a record of an individual bug’s category and status.
The system must log reporter name and e-mail	In case the bug description is vague or some other additional information is needed, the person who reported the error should be contacted.
New bug entries must be reported via e-mail	Once a bug is reported, e-mail should be sent to the appropriate e-mail address or a group of e-mail addresses.
The bug reports should be easily printable	Nielsen [4, p.95] underlines that under certain circumstances, traditional printouts are still very valuable, for example, as handouts in meetings.

4. Planning

The first step in planning the system was to do research on the available solutions on the market. If a suitable system was found, it could be possible to modify the system to meet the requirements set. Once the decision was made, the technologies and their interactions were chosen and planned. The final step comprises the functional and visual design of the user interfaces.

4.1. Available alternatives

Bug tracking is commonly considered essential in software development endeavors. Joel Spolsky identifies the reason in his web log article [6]: "If you are developing code, even on a team of one, without an organized database listing all known bugs in the code, you are simply going to ship low quality code."

Since bug tracking is an everyday practice, there are several tools on the market. Reinventing the wheel is not necessarily sensible, so it is rational to evaluate if one of these ready-made products or parts of them could be used. These offerings can be roughly divided into two groups: commercial products and free publicly licensed products.

Commercial options

There is a wide range of commercial bug tracking tools on the market. Both the feature and pricing range of these products is considerable. According to Softwise Company's comparison chart (APPENDIX D) the license fees range from USD $100 to USD $1950 and beyond. The chart may be somewhat biased since it is used in the company's marketing material for their PR-Tracker product.

The expensive high-end products such as Rational ClearQuest are highly specialized products with an excessive feature list meant for complex application development. It is clear that Morning had no need for such a tool. In addition, the commercial products might be difficult to modify to meet all the requirements that were set in the requirements phase.

Publicly licensed options

The GNU Free Software directory [7] alone lists 19 different bug tracking tools (February 2004) available for use. Most of these applications are licensed under the GNU GPL. The license [8] states the following: “You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program”. Program refers to an application, etc. licensed under the GNU GPL. The terms allow the software to be modified (with some conditions) for commercial purposes.

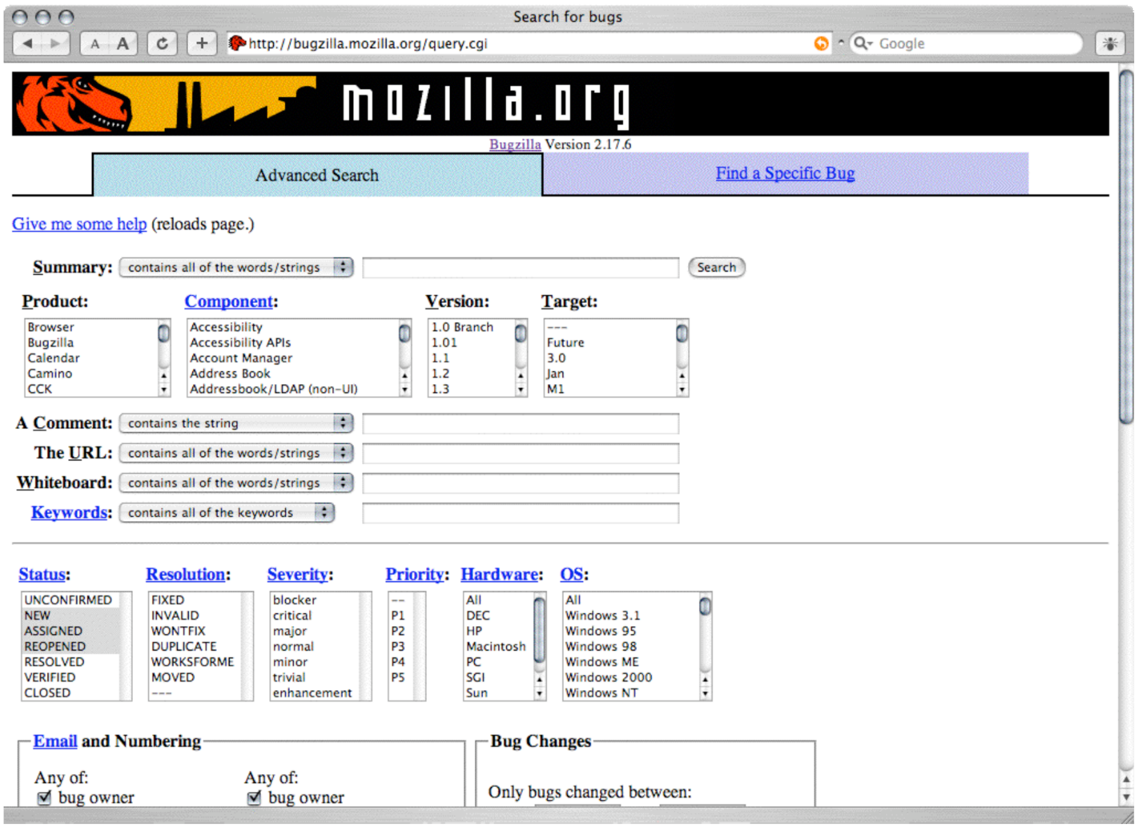
The public license makes the listed applications ideal for modifying them to meet the requirements of the project. The plethora of free alternatives makes it difficult to choose a particular one. Due to its popularity, the choice was to closely examine the Mozilla organization’s Bugzilla system. It is used in both open source and commercial projects. The Mozilla web site states it is used in hundreds of projects in Fortune 500 companies. [9]

Bugzilla is based on a MySQL database and Perl scripting language. It features a browser-based interface for administration and reporting. Development of the system began in 1998 and it is now at version 2.17.6 (February 2004). Efforts to further improve the system remain strong and updates are released regularly.

The system is meant to be a general-purpose tool for debugging applications of different types such as:

- Desktop applications
- Real-time systems
- Web applications

This results in the system having countless features that are not needed in web development projects. The standard reporting form, for example, has fields for detailing the exact setup (hardware and software) of your system. The query interface showing some of the options is shown in Image 2.


Search for bugs

http://bugzilla.mozilla.org/query.cgi

Bugzilla Version 2.17.6

Advanced Search | Find a Specific Bug

Give me some help (reloads page.)

Summary: contains all of the words/strings [Search]

Product:	Component:	Version:	Target:
Browser	Accessibility	1.0 Branch	---
Bugzilla	Accessibility APIs	1.01	Future
Calendar	Account Manager	1.1	3.0
Camino	Address Book	1.2	Jan
CCK	Addressbook/LDAP (non-UI)	1.3	M1

A Comment: contains the string

The URL: contains all of the words/strings

Whiteboard: contains all of the words/strings

Keywords: contains all of the keywords

Status:	Resolution:	Severity:	Priority:	Hardware:	OS:
UNCONFIRMED	FIXED	blocker	--	All	All
NEW	INVALID	critical	P1	DEC	Windows 3.1
ASSIGNED	WONTFIX	major	P2	HP	Windows 95
REOPENED	DUPLICATE	normal	P3	Macintosh	Windows 98
RESOLVED	WORKSFORME	minor	P4	PC	Windows ME
VERIFIED	MOVED	trivial	P5	SGI	Windows 2000
CLOSED	---	enhancement		Sun	Windows NT

Email and Numbering

Any of: ☑ bug owner

Any of: ☑ bug owner

Bug Changes

Only bugs changed between:

Image 2 - Bugzilla bug query interface

Conclusion

After conducting an evaluation of both commercial and publicly available options, it was decided that the best option was to build the system from ground up. Commercial systems were ruled out due to pricing issues and potential problems and effort required to modify them to reach the desired functionality. The open source option, Bugzilla, was abandoned for the latter reason.

4.2. System specifications

4.2.1. Basic dataflow

The system can be broken down to the following elements and actions:

- Data input (reporting)
- Storage
- Administration
- Correction reporting

All the bug data in the system is a result of a user submitting a report. Depending on the reporting method some automatic data may also be sent. This data is stored in a database. The gathered data can be viewed and managed with the available administration tools. Once a bug is solved, the status is set as closed. At this point, an automatic correction report may be sent to the user that reported it.

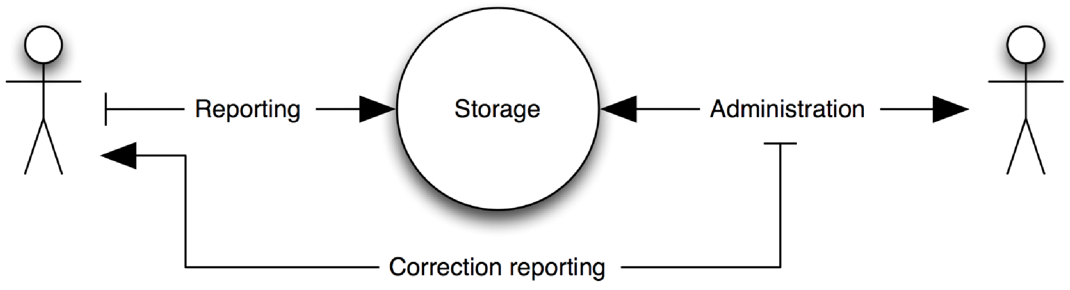


Figure 1 - Basic dataflow in the Marmalade system

4.2.2. Architecture

As shown in Figure 1 the basic dataflow of the system is quite simple. There are two users interacting on single data storage. In reality, the flow is more complicated and has various forms. Figure 2 shows all the interacting components of the system.

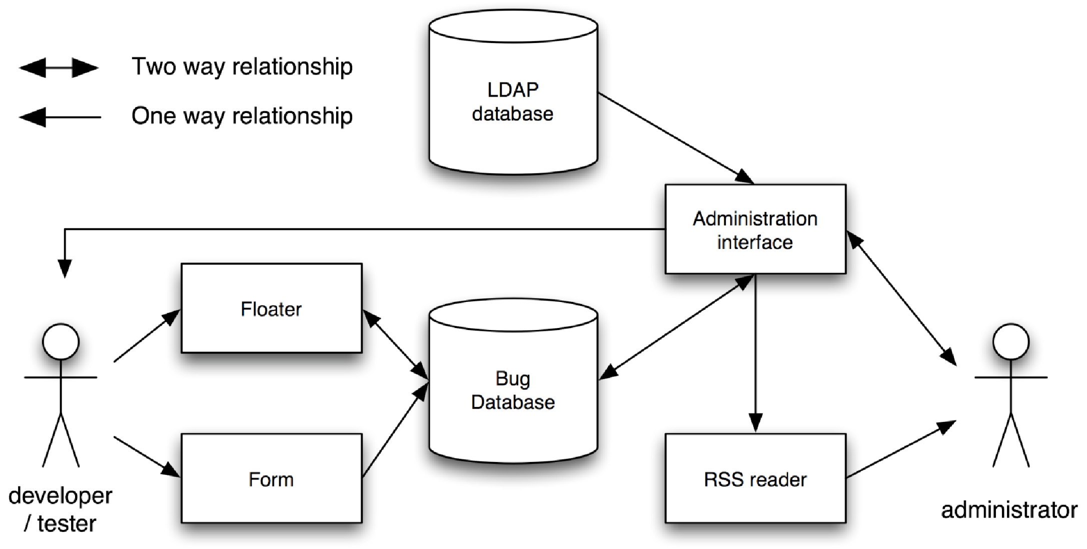


Figure 2 - The components of the Marmalade 1.0 system and their interactions

The functions of the components are:

- The floater and form are alternative methods for users to input form data. The floater is the preferred method, due to the additional data it sends.

- The bug database is the central storage for data. It contains information about individual bugs, projects and their relations.

- The administration interface is used to administer the data in the system. It is the only component to modify the bug database. It also sends correction reports to the bug reporters.

- The LDAP database is used by the administration interface to authenticate users.

- The RSS reader is an independent desktop application that is used to browse the data in the database. The required data is generated by the administration interface.

4.2.3. Server platform

The choice for the server platform was the so-called LAMP platform. It is one of the most common platforms for web applications on the Internet. The LAMP platform consists of the following four components:

- Linux (operating system)
- Apache (web server)
- MySQL (database)
- PHP (scripting language)

The choice to use the LAMP platform was made for the following reasons:

- The company already had a suitable server
- Proven technology
- A good environment for creating database driven web applications

Since Apache, MySQL and PHP are available for many platforms, it is possible to transfer the application to a platform running another operating system such as another UNIX like system or Microsoft Windows. The development of the Marmalade 1.0 system was made on a laptop computer running Mac OS X, Apache, MySQL and PHP.

4.2.4. Database

MySQL is an open source database system originally developed by Michael Widenius in 1995. It is licensed under the GNU public license, but commercial license options also exist. The system is available for a wide array of platforms and uses the common Structured Query Language (SQL) for executing the database functions.

Today MySQL is the most popular open source database in the world. The current version is 4.0.18. The next major version, 5.0, has already reached alpha status. PHP has had native MySQL support since version 3.0. The available functions make development of database driven web applications effortless.

The table structure of the Marmalade 1.0 database is shown in Figure 3.

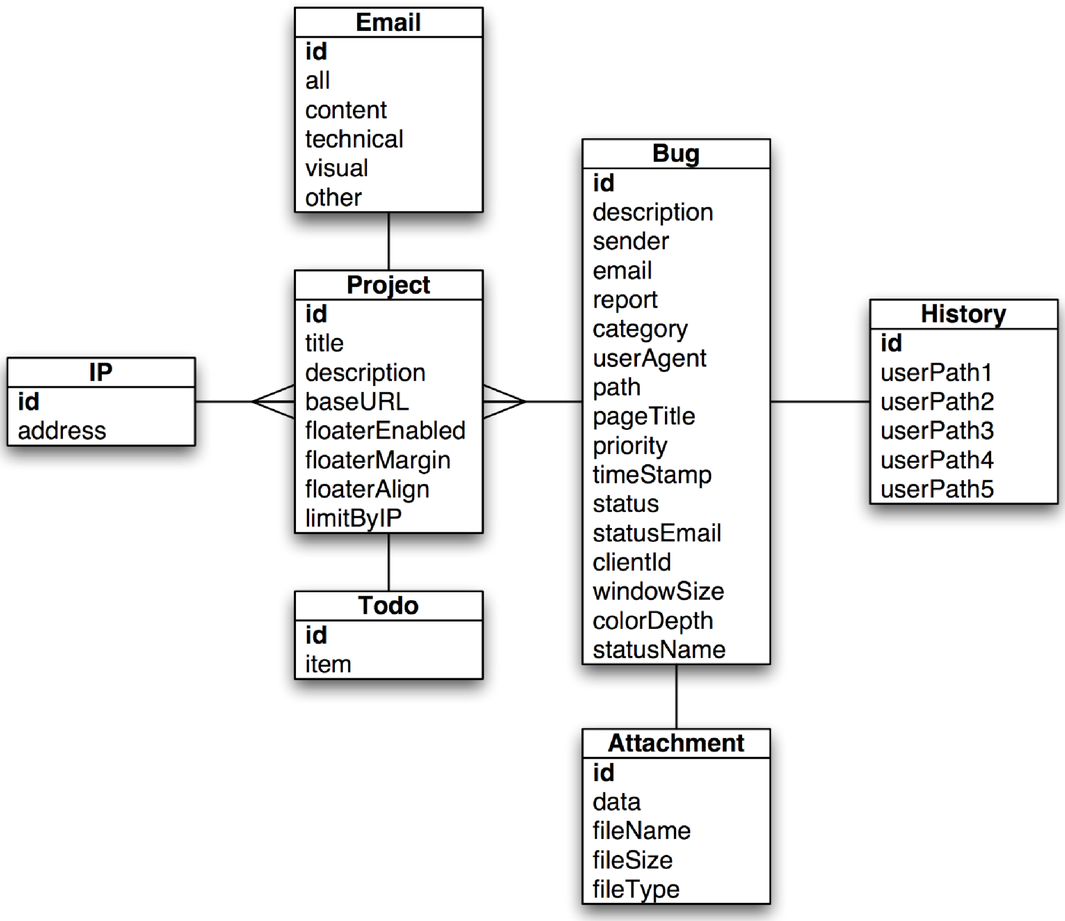


Figure 3 - The database structure

4.2.5. LDAP authentication

Lightweight Directory Access Protocol (LDAP) is a protocol used to locate the following resources on a computer network:

- Organizations
- Individuals
- Files
- Devices

Authentication to these resources can also be set. LDAP can be used over a private network or a public one such as the Internet. It could be considered as a version of DAP (Directory Access Protocol) with a reduced feature set. There are currently multiple commercial and open source products utilizing LDAP.

The current implementation (LDAPv3) has been developed by The Internet Engineering Task Force. The IETF web site describes the organization as follows: "The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet." [10]. LDAPv3 was released in 1997. Efforts to improve and develop the protocol remain strong.

On the Morning LAN (Local Area Network), user information is stored on a Microsoft Windows Active Directory server. The stored data is used to give users access to the network domain, fileserver directories and Microsoft Outlook e-mail accounts. The Active Directory server is not accessible from the Internet (shown in Figure 4).

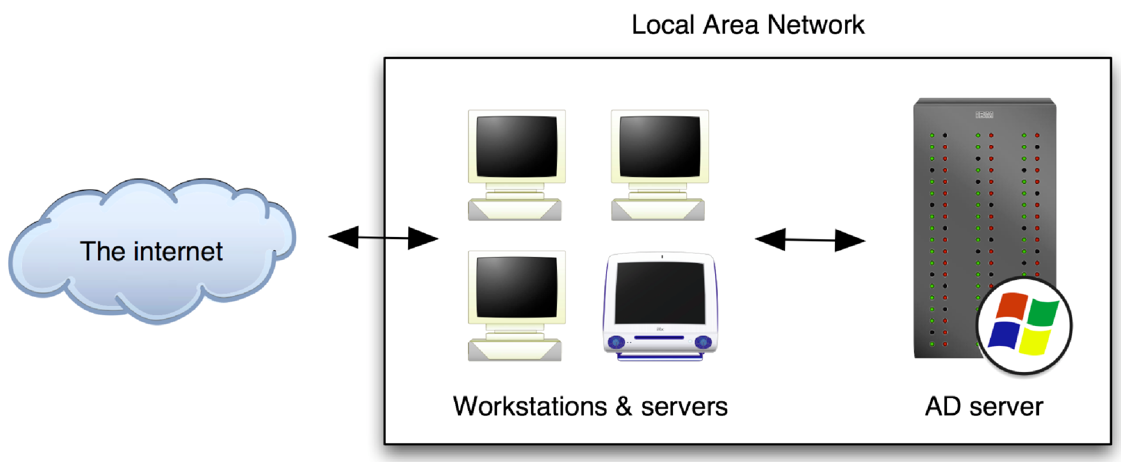


Figure 4 – Basic network structure

PHP has had native support for LDAP databases since version 3.0. There are functions for connecting, performing queries and processing result sets. These ready-made functions enable simple development of LDAP-aware web applications using PHP.

LDAP authentication in PHP applications had already been used at Morning. This meant the existing authentication module could be used to add LDAP authentication to Marmalade. At first, the authentication will only authenticate users, but not allow any personalization. With LDAP the authentication module in place, it will be convenient to add such features in the future.

4.2.6. RSS feed

RSS is a technology originally created by Netscape for distributing and syndicating content for their web portal service in 1999. The company has since lost interest in developing the technology, but the refinement process was picked up by Userland Software.

RSS was originally targeted for news distribution, but xml.com [11] states: “Pretty much anything that can be broken down into discrete items can be syndicated via RSS: the "recent changes" page of a wiki, a changelog of CVS checkins, even the revision history of a book.”

Mark Nottingham describes RSS as follows: “RSS is an XML-based format that allows the syndication of lists of hyperlinks, along with other information, or metadata, that helps viewers decide whether they want to follow the link.” [12]

RSS is a mass of XML data. This data is read and interpreted by an application. The application can be a desktop application displaying the data to the user or it can be, for example, a web application that extracts parts of the feed and displays them on a web site. RSS feeds can be accessed through standard HTTP connections.

There are multiple versions of the RSS standard in use. The most popular versions today are 0.9.x and 1.0. Both versions are still actively developed by a number of organizations. The 2.0 specification has also been released. All versions are based on the original Netscape specifications.

Due to their simplicity, 0.9.x versions remain as the best option for simple content distribution and syndication needs. The Marmalade system will use RSS version 0.91. The basic structure of an RSS 0.91 feed is shown in Example 1.

```
<?xml version="1.0"?>
<!DOCTYPE rss PUBLIC "-//Netscape Communications//DTD RSS
0.91//EN" "http://my.netscape.com/publish/formats/rss-0.91.dtd">
    <rss version="0.91">
    <channel>
    <title></title>
    <link></link>
    <description> </description>
    <language></language>
        <item>
        <title></title>
        <description></description>
        </item>
    </channel>
</rss>
```

Example 1 - The XML structure of RSS version 0.91

The Marmalade 1.0 system will give users access to bug reports using RSS feeds. These feeds can be read using an RSS reader application. The timed refresh feature of the applications offers an automatic solution to the bug tracking requirement stated by Thomas J. Shelford: “Everyone on the team should have access to the bug database and should be checking it frequently.” [13, p. 192]

4.3. Interfaces

The Marmalade 1.0 system has two separate user interfaces:

- Administration interface
- Reporting floater interface

The planning phase contains the functional and visual design of the interfaces. The initial planning was done using mockups, which demonstrate the location and functions of the elements: "Mockups are primarily single-page static representations of the design space" [2, p. 216]. The finalizing of the layouts was done during the implementation phase.

4.3.1. Administration interface

The navigational structure of the administration interface was set at the beginning of the planning phase. The interface has seven basic locations. They are shown in Figure 5.

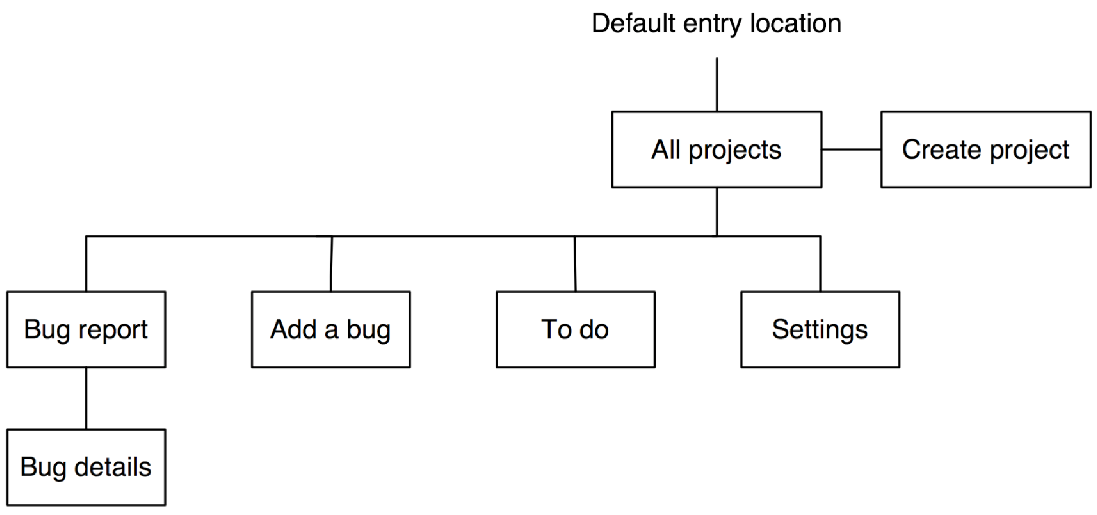


Figure 5 - Navigational structure of the administration interface

For consistency, each location uses the same basic layout template: “The use of a page template allows you to establish a consistent and simple page structure throughout the web site.” [2, p. 184]

The template has seven distinct layout elements. The elements are shown in Table 2.

Table 2 - Administration interface elements

ID	**Element**	**Description**
1	Logo	Holds the logo for quick visual identification.
2	Location header	Displays the function of the page.
3	Date and time	Acts as a time stamp.
4	Primary navigation	Holds the links for the major functions in the system. It is always visible and has two modes. The modes and links are shown in Figure 7. The selected item is indicated by changing the background color of the link.
5	Secondary navigation	Provides access to additional functions. It has two modes, which are shown in Figure 8.
6	Content area	Displays the content.
7	Footer area	Contains links to additional functions and contact information.

The placement of the seven elements in the design space is shown in Figure 6.

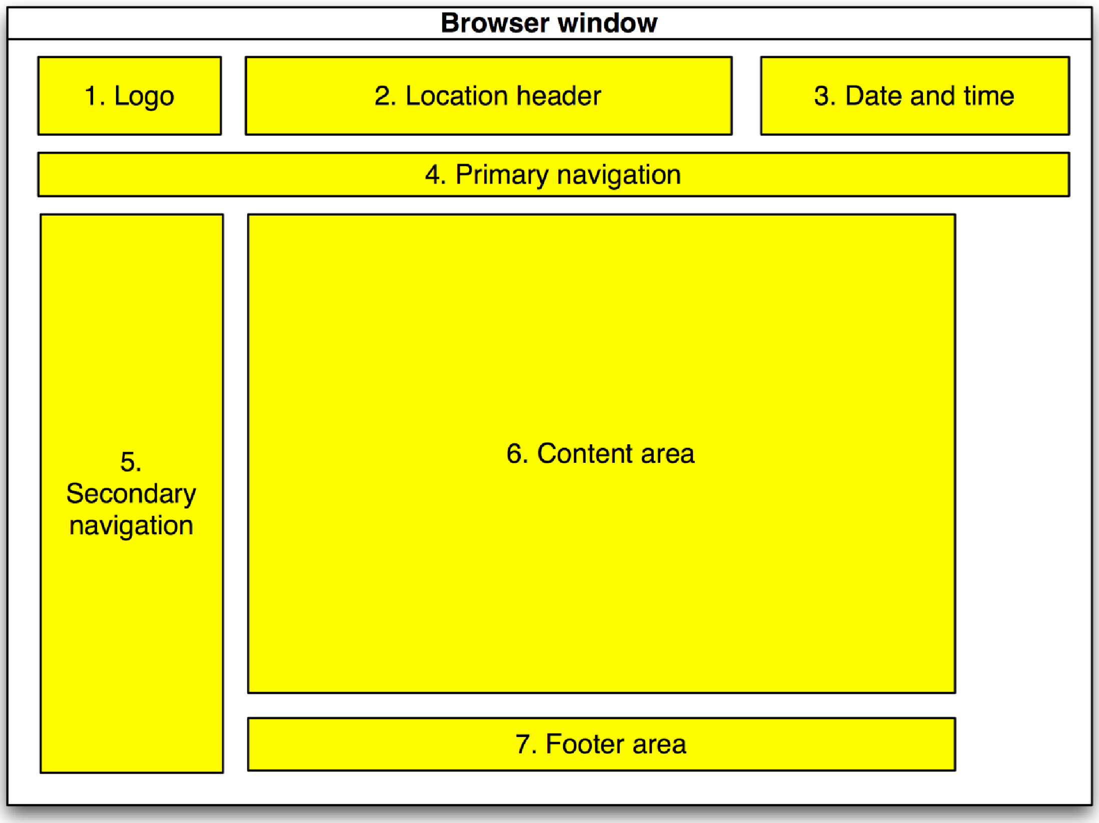


Figure 6 - Administration interface elements

The layouts of the navigation elements 4 and 5 are shown in Figures 7 and 8.

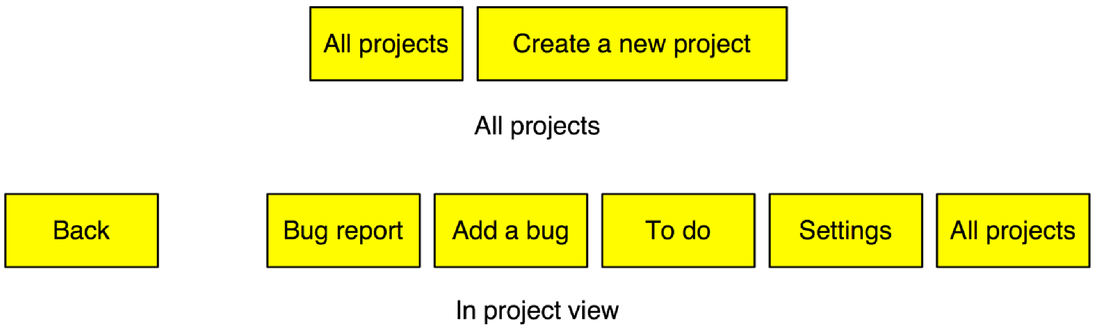
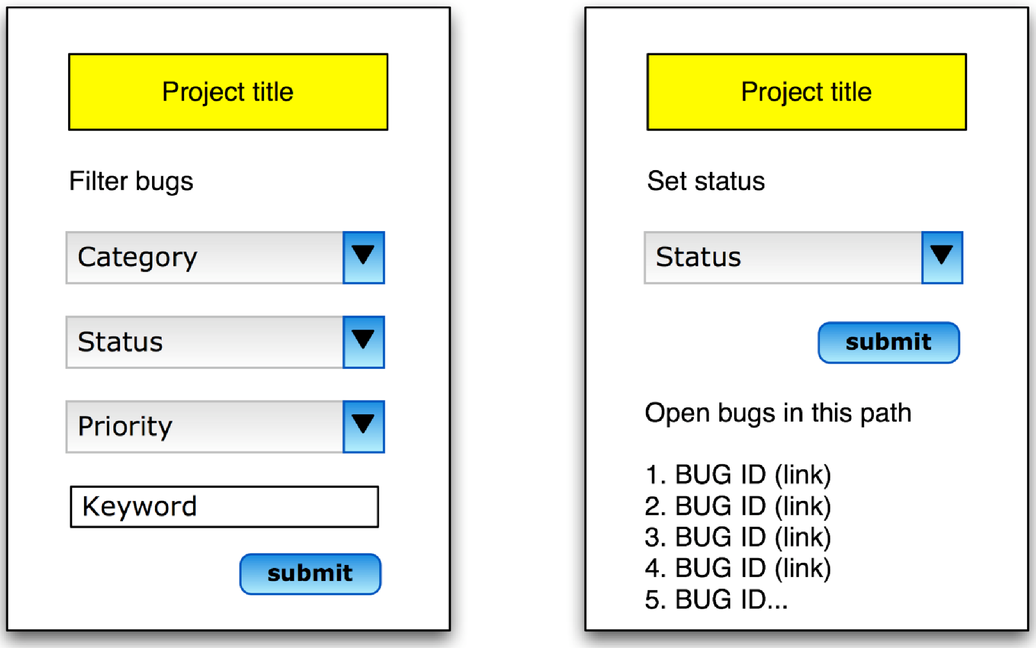


Figure 7 - Primary navigation elements



Bug report Bug details

Figure 8 - Secondary navigation elements

The navigational items use distinctive color combinations of black, white and yellow: "Highly contrasted elements grab the users' attention, whereas more subdued contrasts require conscious effort to be noticed" [2, p. 188]. Red, yellow and green colors were used to indicate the status and priority of a specific bug.

In general, the visual design of the interface was kept simple: "A truly elegant design reduces the page to its required elements. This allows each element to be intimately tied to its message and increases the page structure perceived by the user." [2, p. 180]

Screen captures of the final interface are shown in APPENDIX A.

4.3.2. Floater interface

The goal was to create an interface that would be easy for the developers to deploy and the users to use: “The bug reporting tool should be easy to use and understand. The goal of any communication tool is to keep everyone on the same page and in tune with each other and the effort at hand.” [13, p. 192]

The floater interface is built using a combination of PHP and JavaScript. The server-side scripting language (PHP) creates a suitable client-side script (JavaScript) that is executed by the browser. This combination of server-side and client-side scripting makes it possible to receive information from an external database without adding server-side scripting to the tracked system.

The floater is displayed to the user as a layer floating in the browser window. The default view (shown in Figure 9) holds the following items:

- Project name
- Report a bug (link)
- View bugs (link)

If the user wishes to report a bug, the first step is to click the view bugs link. This opens a browser window with a listing of bugs logged for the specific page. The bug listing window is shown in Figure 10.

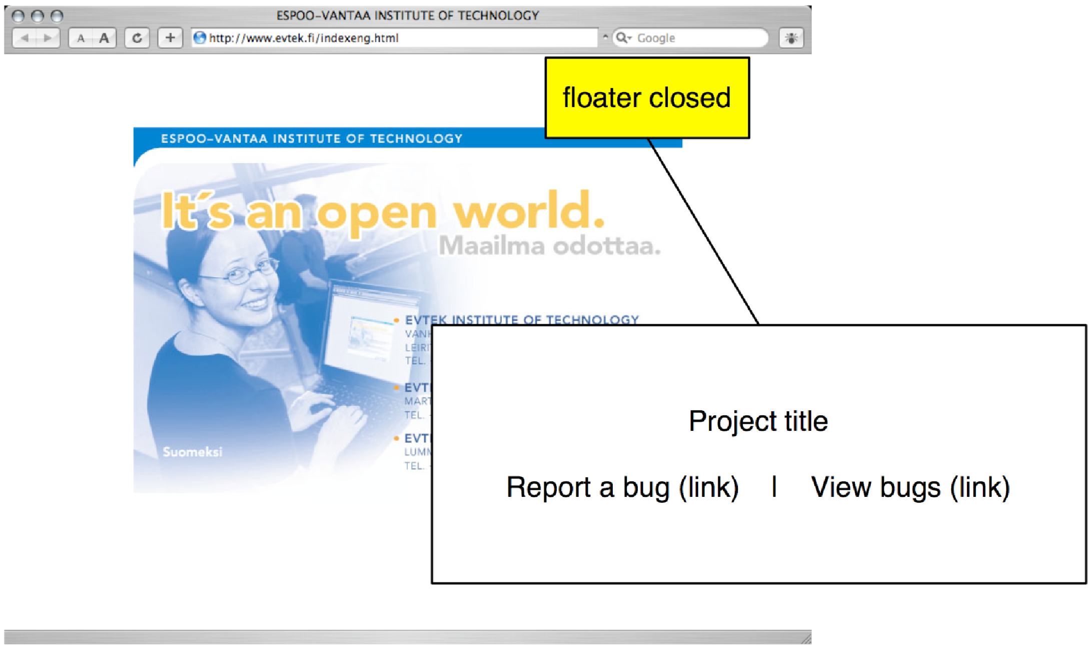

ESPOO-VANTAA INSTITUTE OF TECHNOLOGY
http://www.evtek.fi/indexeng.html
floater closed
It´s an open world.
Maailma odottaa.
Suomeksi
EVTEK INSTITUTE OF TECHNOLOGY
Project title
Report a bug (link) | View bugs (link)

Figure 9 - Floater interface mockup (floater closed)

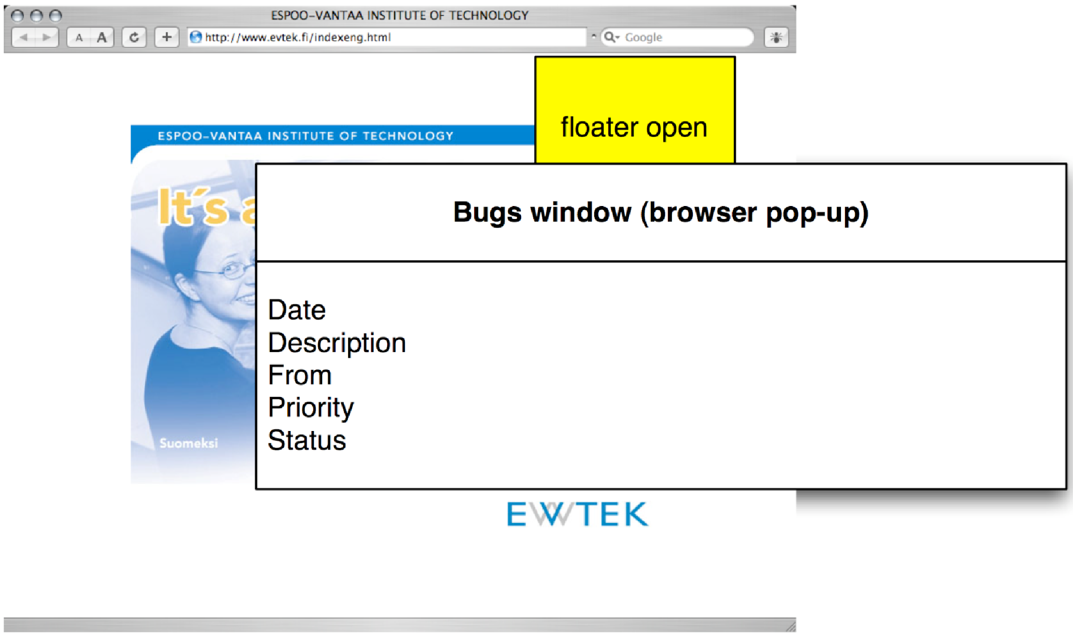

ESPOO-VANTAA INSTITUTE OF TECHNOLOGY
http://www.evtek.fi/indexeng.html
floater open
Bugs window (browser pop-up)
Date
Description
From
Priority
Status
EVTEK

Figure 10 - Floater interface mockup (view bugs)

If the bug has not yet been reported, the user clicks the report a bug link. It opens a reporting form (shown in Figure 11). If the user decides not to report the bug, the close link is used to return the floater to the default state.

If the user tries to submit the form without the required fields, an alert box is shown (shown in Figure 12).

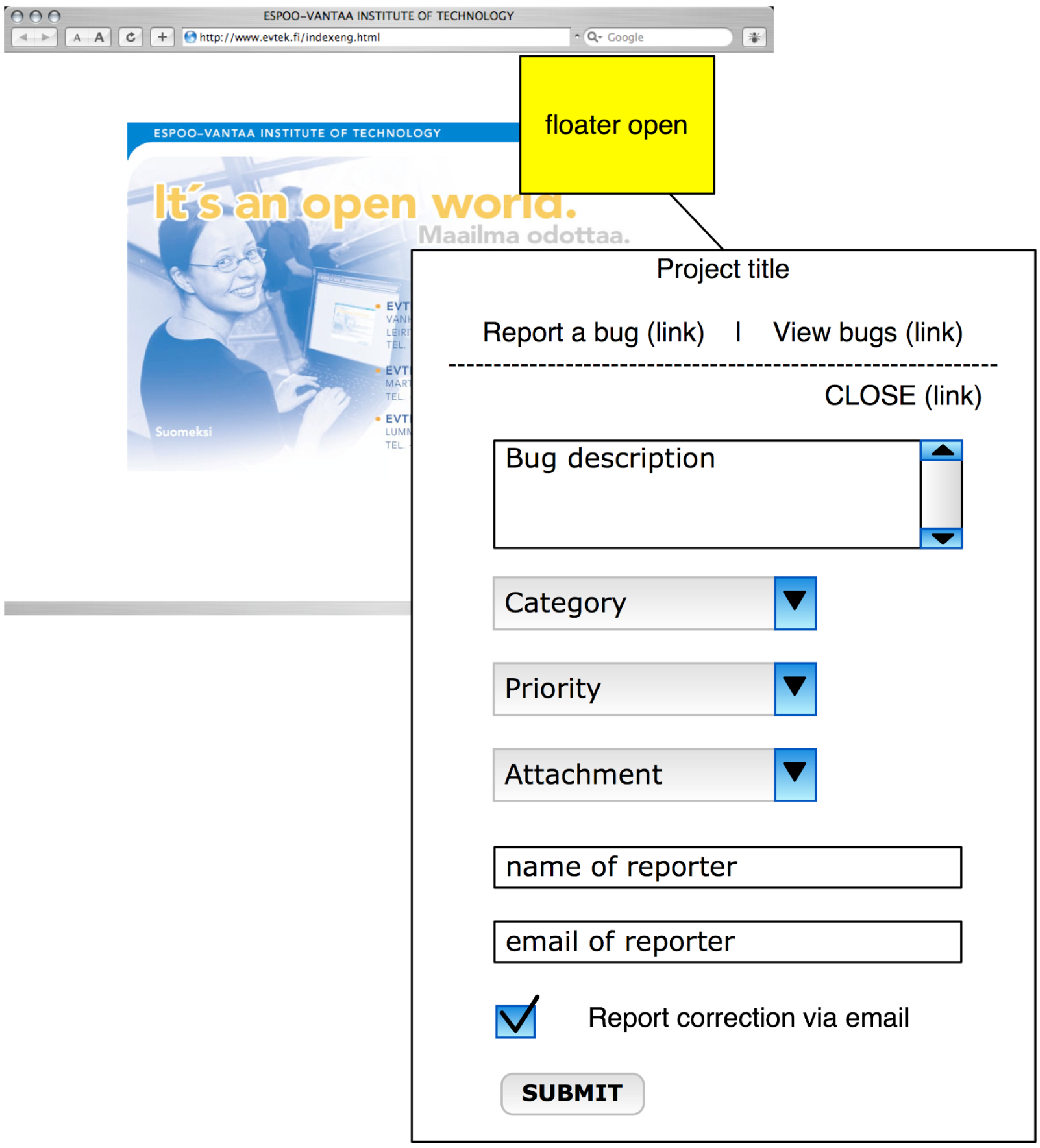
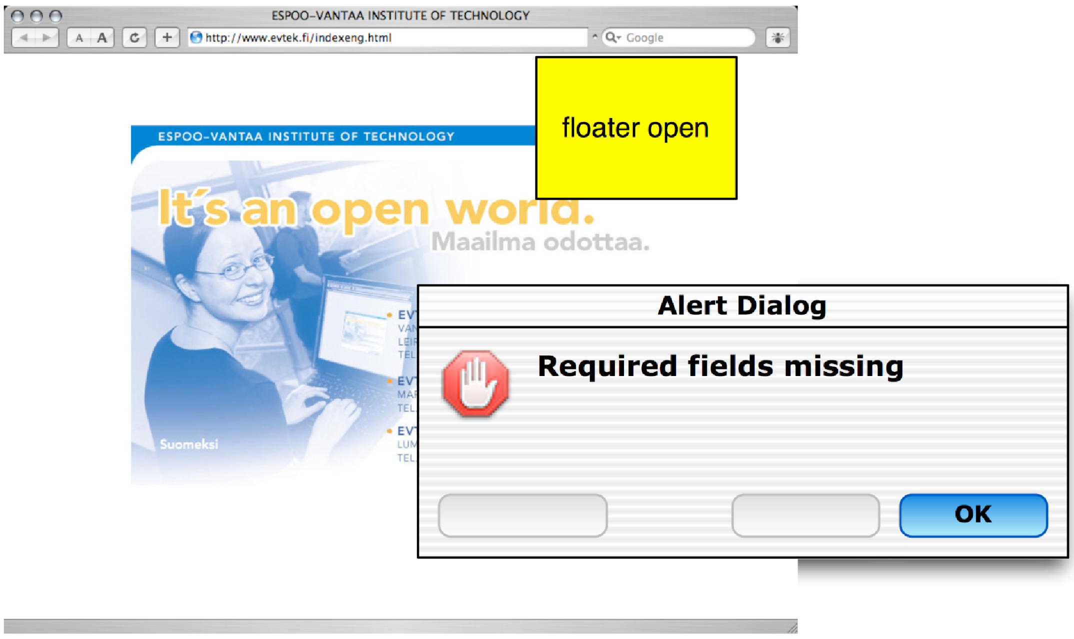

ESPOO-VANTAA INSTITUTE OF TECHNOLOGY
http://www.evtek.fi/indexeng.html
Google
ESPOO-VANTAA INSTITUTE OF TECHNOLOGY
It´s an open world.
Maailma odottaa.
Suomeksi
floater open
Project title
Report a bug (link) | View bugs (link)
CLOSE (link)
Bug description
Category
Priority
Attachment
name of reporter
email of reporter
Report correction via email
SUBMIT

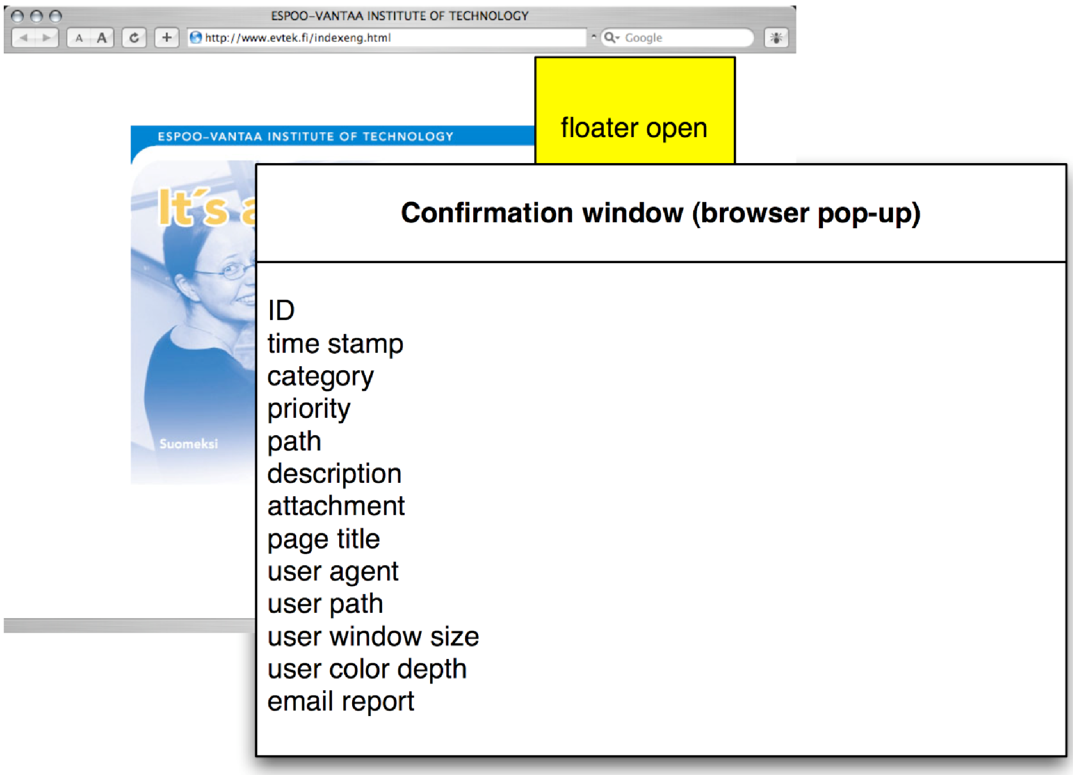
Figure 11 - Floater interface mockup (floater opened)


ESPOO-VANTAA INSTITUTE OF TECHNOLOGY
http://www.evtek.fi/indexeng.html
Google
floater open
ESPOO-VANTAA INSTITUTE OF TECHNOLOGY
It´s an open world.
Maailma odottaa.
Suomeksi
Alert Dialog
OK

Required fields missing

Figure 12 - Floater interface mockup (missing fields)

If a user successfully submits a report, a confirmation is opened (shown in Figure 13).


ESPOO-VANTAA INSTITUTE OF TECHNOLOGY
http://www.evtek.fi/indexeng.html
Google
floater open
ESPOO-VANTAA INSTITUTE OF TECHNOLOGY
Suomeksi

Confirmation window (browser pop-up)

ID
time stamp
category
priority
path
description
attachment
page title
user agent
user path
user window size
user color depth
email report

Figure 13 - Floater interface mockup (successful submit)

The basic flow of the floater application is shown in Figure 14.

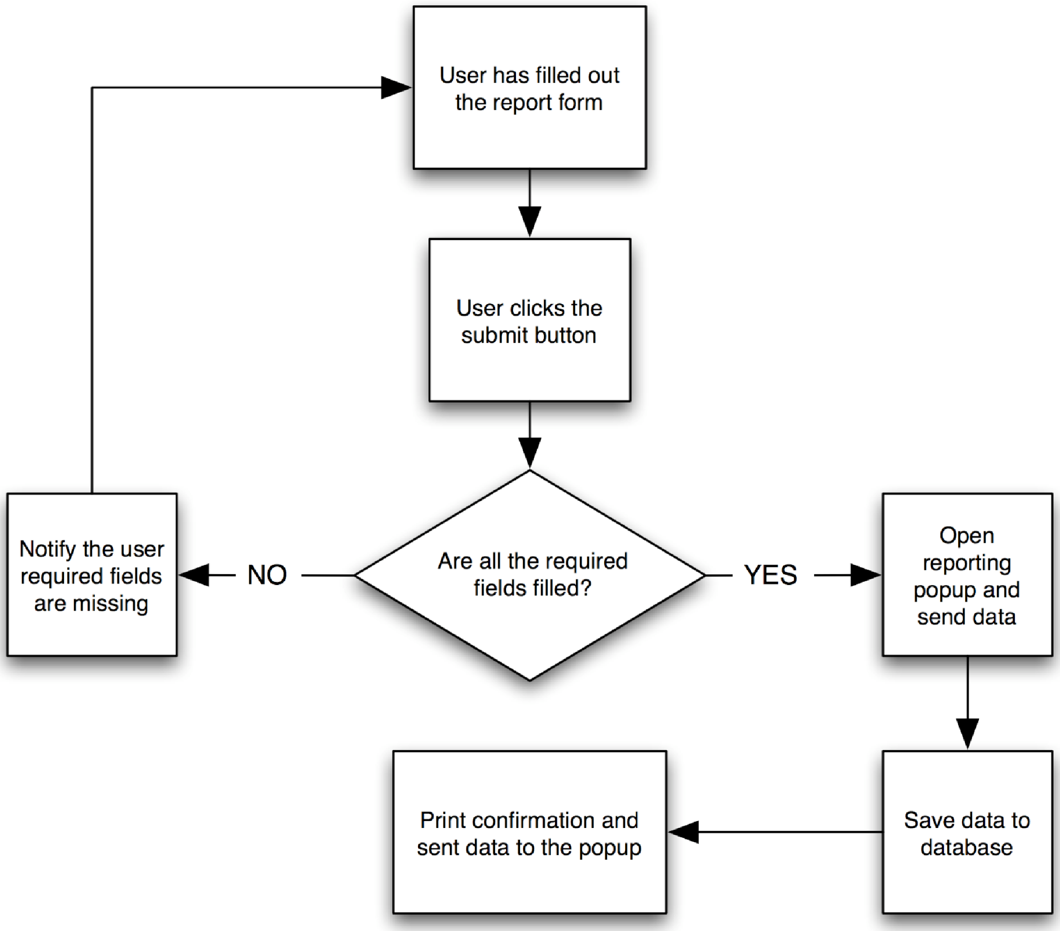

User has filled out the report form
User clicks the submit button
Are all the required fields filled?
NO
Notify the user required fields are missing
YES
Open reporting popup and send data
Save data to database
Print confirmation and sent data to the popup

Figure 14 – Simplified floater process flow

The visual design closely resembles the administration interface. The aim was to keep the footprint of the floater element as small as possible.

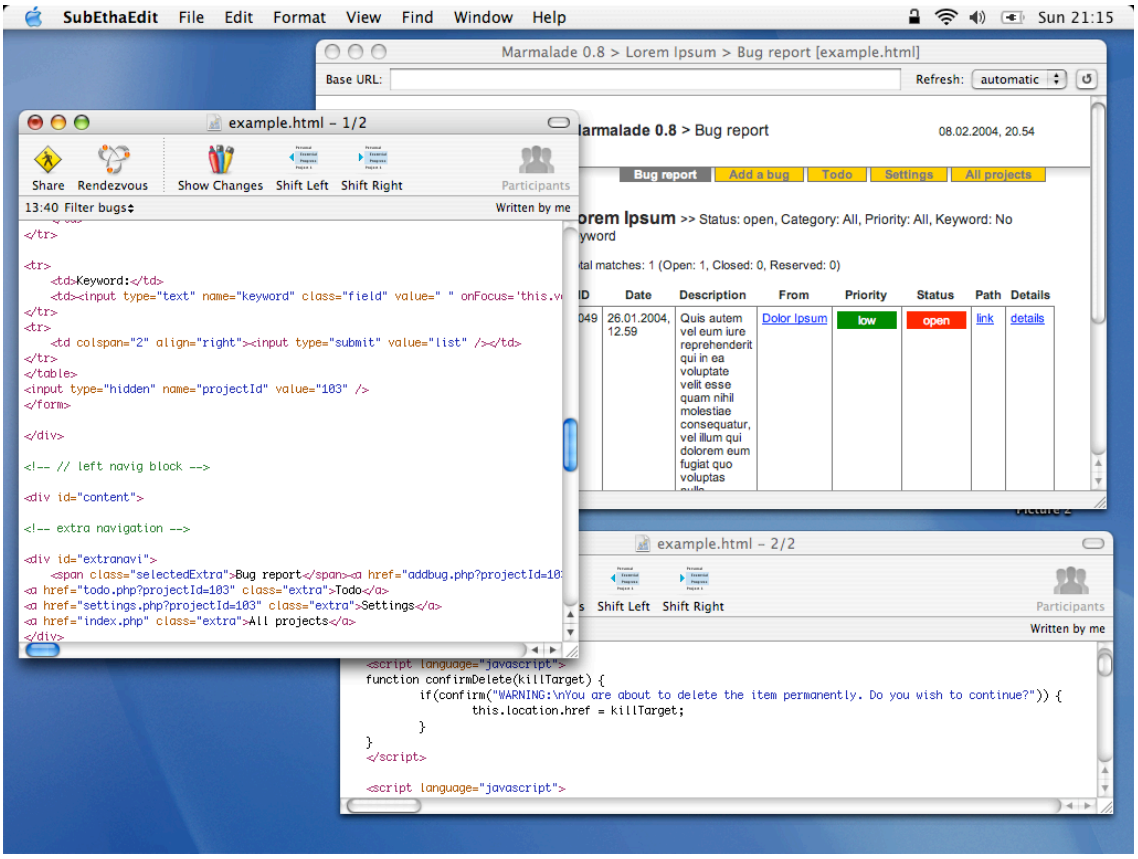
Screen captures of the final interface are shown in APPENDIX B.

5. Implementation

The construction process was done using a development platform. Features were tested as they were built. The final step was to transfer the system to the production platform.

5.1. Development platform and tools

The development platform was an Apple iBook with Mac OS X version 10.3.x. The web server was the default Apache (version 1.3.29) installation. PHP (version 4.3.4) and MySQL (version 4.0.16) were installed using Marc Liyanage's packages (from http://www.entropy.ch). In addition to standard UNIX command line tools, some graphical applications were used. The development work was done using the SubEthaEdit (version 1.1.5) text editor, which is integrated with the XHTML-rendering engine for convenient rendering experimenting. A screen capture is shown in Image 3.



```
</tr>

<tr>
    <td>Keyword:</td>
    <td><input type="text" name="keyword" class="field" value=" " onFocus='this.v
</tr>
<tr>
    <td colspan="2" align="right"><input type="submit" value="list" /></td>
</tr>
</table>
<input type="hidden" name="projectId" value="103" />
</form>

</div>

<!-- // left navig block -->

<div id="content">

<!-- extra navigation -->

<div id="extranavi">
    <span class="selectedExtra">Bug report</span><a href="addbug.php?projectId=10
<a href="todo.php?projectId=103" class="extra">Todo</a>
<a href="settings.php?projectId=103" class="extra">Settings</a>
<a href="index.php" class="extra">All projects</a>
</div>
```

armalade 0.8 > Bug report 08.02.2004, 20.54

Bug report | Add a bug | Todo | Settings | All projects

orem Ipsum >> Status: open, Category: All, Priority: All, Keyword: No yword

tal matches: 1 (Open: 1, Closed: 0, Reserved: 0)

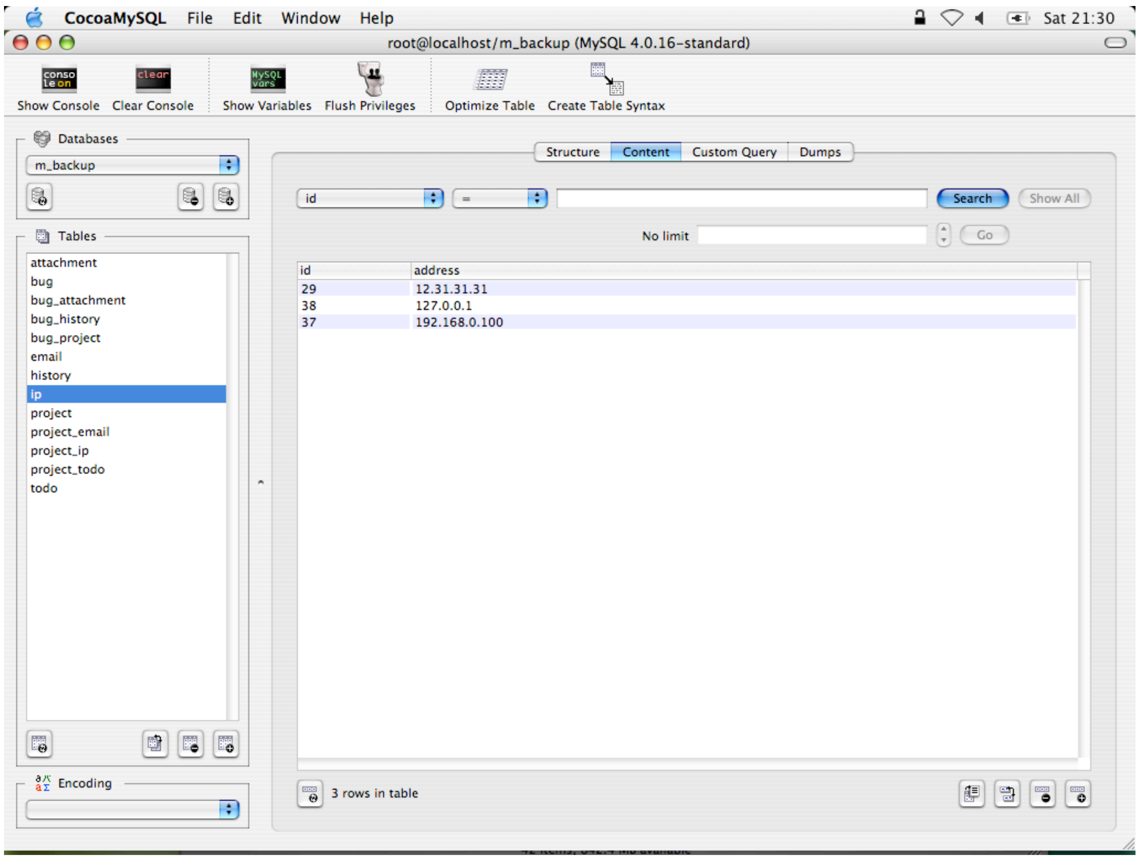
ID	Date	Description	From	Priority	Status	Path	Details
049	26.01.2004, 12.59	Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas	Dolor Ipsum	low	open	link	details

```
<script language="javascript">
function confirmDelete(killTarget) {
        if(confirm("WARNING:\nYou are about to delete the item permanently. Do you wish to continue?")) {
                this.location.href = killTarget;
        }
}
</script>

<script language="javascript">
```

Image 3 - A typical view to the SubEthaEdit editor

The database queries were built (and tested) using CocoaMySQL (version 0.5), a graphical interface for the MySQL database. A screen capture is shown in Image 4.



id	address
29	12.31.31.31
38	127.0.0.1
37	192.168.0.100

3 rows in table

Image 4 - The interface of the CocoaMySQL application

In addition to the development platform, different setups were used to test the interfaces and functionality of the product. A PC-compatible desktop computer (700 MHz) with Microsoft Windows XP and Gentoo LiveCD (version 1.4) Linux distribution was the common testing setup. Both operating systems had multiple browsers from different vendors installed. The range of different combinations is shown in Table 3.

Table 3 - Operating system and browser configurations used in floater testing

Operating system	**Browser**			
	Internet Explorer (Version 6.x.x)	Mozilla Firebird (Version 0.7.x)	Opera (Version 7.x.x)	Safari (Version 1.1.x)
Mac OS X		X		X
Linux		X	X	
Windows	X	X	X	

5.2. Back-end and administration interface

The administration interface was built according to the common three-tier model. The model is comprised of three basic components:

- Presentation tier (web browser)
- Business logic tier (web server + server-side processing)
- Services tier (database or other back-end services)

The tiers are independent and only communicate to the adjacent tier. Physically the business logic and services tier can run on the same server, but as independent application processes. The browser communicates to the web server through a network, such as the Internet or a local area network. The flow of a typical web application:

- The browser sends a request
- The web server receives the request and starts processing
- The server-side process requests data from the database
- The database returns the requested data
- The web server returns the processed data to the browser

The presentation tier

In the case of web applications, browsers send a HTTP request to the web server, which returns a XHTML file. The file contains the data as well as additional layout information. The browser renders the received file and outputs the data. This is usually done by displaying it on a computer screen, but browsers for the hearing impaired, for example, output it as speech.

The presentation layer of the Marmalade 1.0 system was built using standard XHTML and an external CSS file. The CSS file is used to set the appearance of XHTML documents. Since the interface has a default layout, the business logic tier was built to use a template file for producing the XHTML output. The template file contains calls for content output functions. An example is shown in Example 2.

```
<!-- primary navigation -->
<div id="primarynavi">
    <?php write_primarynavi(); ?>
</div>
<!-- // primary navigation -->
```

Example 2 – Writing the primary navigation to the template by calling the function

By utilizing a template-based business logic tier and external style sheets the layout of the application can be changed conveniently. By modifying the template and the CSS file, each page in the system is affected.

Modern browsers allow applying a different style sheet for a page when the page is printed. An example of a printed bug report page is shown in APPENDIX A.

The business logic tier

The business logic tier comprises the Apache web server and PHP scripting language. Once the server has received a request, it is directed to the PHP engine. The engine processes the application and connects to the service tier as needed. Once the processing is completed, the data is sent to the requester.

The system is built using a number of files that have a specific function. The structure of each file (structure shown in APPENDIX F) in the system is identical. The first step in the application is to make sure the user has the right to access the content. This is done by including the ldap.php file. If the user is not valid or has not yet logged onto the system writes a login form and stops the execution.

If the user is valid and execution is not aborted, the system fetches the config.php file that holds basic system parameters such as the database access details and the base URL of the system. The next step is to include the functions.php file. This file holds the functions for writing the navigation and other standard elements that are processed similarly from page to page.

The write_content function is unique to each page and is used to generate the data to be displayed on the content area. This function typically contains calls to the database according to the parameters given in the URL. The write_content function is called in the template file. The final step is including the template file. The functions called in the template file are executed and appropriate output is generated.

The typical function for a page is to view data. The todo.php, for example, has more functions as users can add and delete items. The identifying of different functions is done by using switch cases. Switch cases allow the program to leap to a certain location in the program code according to a parameter. In the Marmalade system the parameter used for switching is given by the value of the XHTML submit element.

An example of the basic structure of a page in the application is shown in Example 3.

```
<?php
require_once("./ldap.php");
require_once("./config.php");
require_once("./functions.php");

switch ($GLOBALS[submit]){

default:

    // write_content function (called in template)
    function write_content(){

    ###########################################
    #                                         #
    # GENERATE AND OUTPUT APPROPRIATE CONTENT #
    #                                         #
    ###########################################

    }
    include("./templates/default.php");

break;

case delete:

    ###########################################
    #                                         #
    # ERASE SELECTED CONTENT AND REDIRECT     #
    #                                         #
    ###########################################

break;
}
?>
```

Example 3 – The basic program structure

The services tier

The Marmalade 1.0 system connects two separate services:

- The MySQL database server
- The LDAP server

The business logic tier communicates with the MySQL database Structured Query Language (SQL). SQL is a standard syntax for adding, removing and displaying the data in a database. PHP has built in functions for connecting to MySQL databases. An extract of a business logic tier content write function is shown in Example 4.

```
$select = "
            SELECT id, title, description
            FROM projects
            ORDER BY id DESC
";
$result = mysql_query($select);
while ($dataColumn = mysql_fetch_array($result)) {
        extract ( $dataColumn );

    ###################################################
    #                                                 #
    # WRITE A ROW OF DATA (ID, TITLE AND DESCRIPTION) #
    #                                                 #
    ###################################################

}
```

Example 4 - Getting data from MySQL using a while loop

The LDAP authentication used by the system was provided through an existing authentication module. This module handles the configuration and connection to the LDAP service. The authentication is implemented by integrating the ready-made module to the flow of the server-application (shown in Example 3). A basic example for testing the LDAP authentication is shown in Example 5.

```
<?php
require_once("auth_ldap_config.php");
require_once("auth_ldap_functions.php");
$user = "user";
$pass = "password";
if(pwpadm_auth_from_ldap($user,$pass)) {
    echo "Authentication ok";
} else {
    echo "Authentication failed";
}
?>
```

Example 5 – Basic LDAP authentication

5.3. Floater interface

The floater component was built using a combination of JavaScript and PHP. It is deployed by including a certain tag in the XHTML of the tracked system. A detailed description of the deployment can be read in chapter 6. Screen captures of the final interface are shown in APPENDIX B.

PHP is used to create a JavaScript file according to the referrer URL (RURL) given by the browser. The exchange of data is shown in Figure 15.

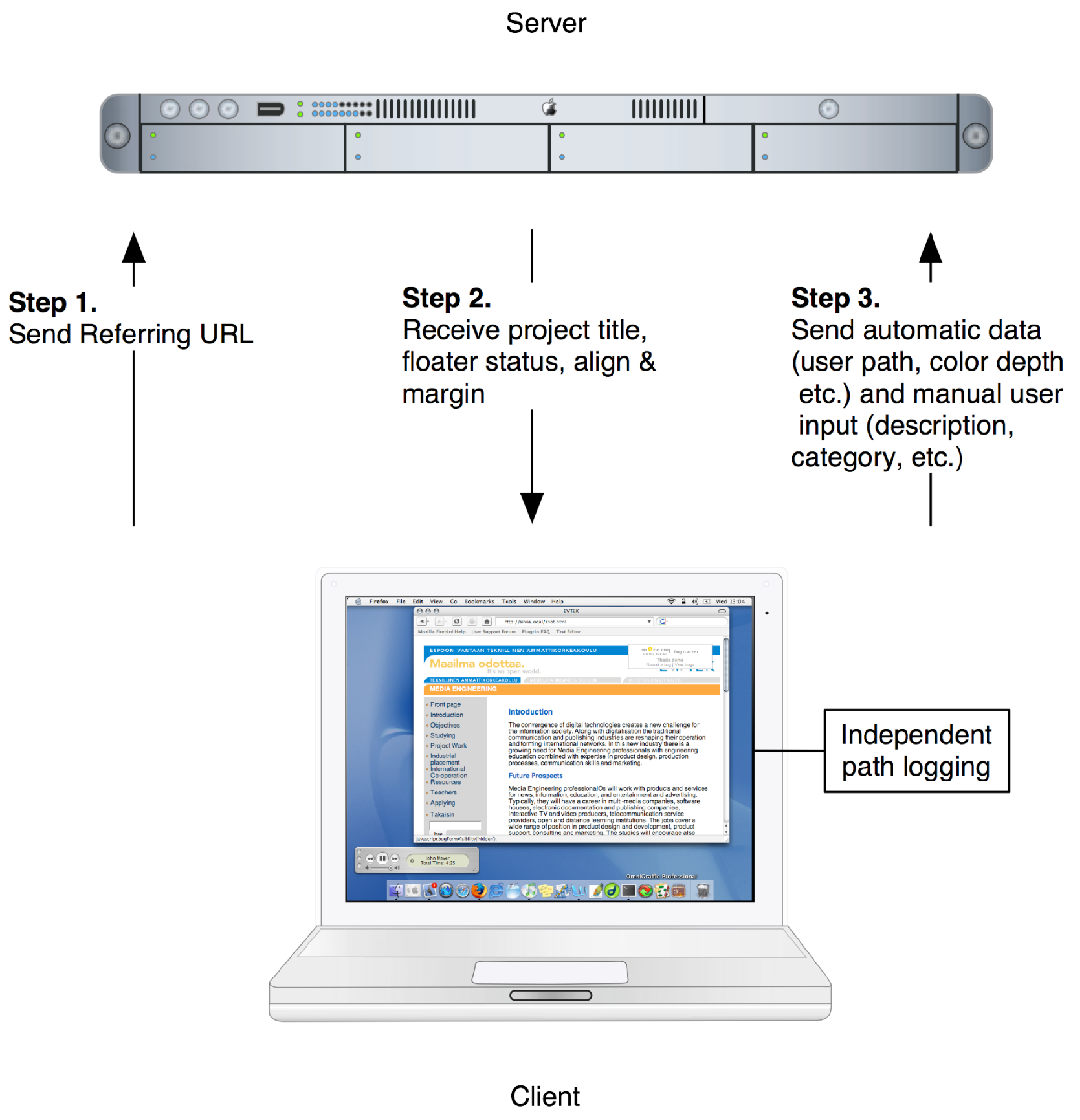

Server
Step 1.
Send Referring URL
Step 2.
Receive project title, floater status, align & margin
Step 3.
Send automatic data (user path, color depth etc.) and manual user input (description, category, etc.)
Independent path logging
Client

Figure 15 - The basic flow of data between the floater client and the server

The referrer URL is the URL of the page containing the reference (the script tag) to the PHP file. The database has a base URL (BURL) field for each project. The BURL is a unique identifier and is used by PHP to identify the project. The RURL is compared to the BURLs in the database. If a match is found the appropriate floater code is returned, otherwise floater code with an error report is returned. The process is illustrated in Figure 16.

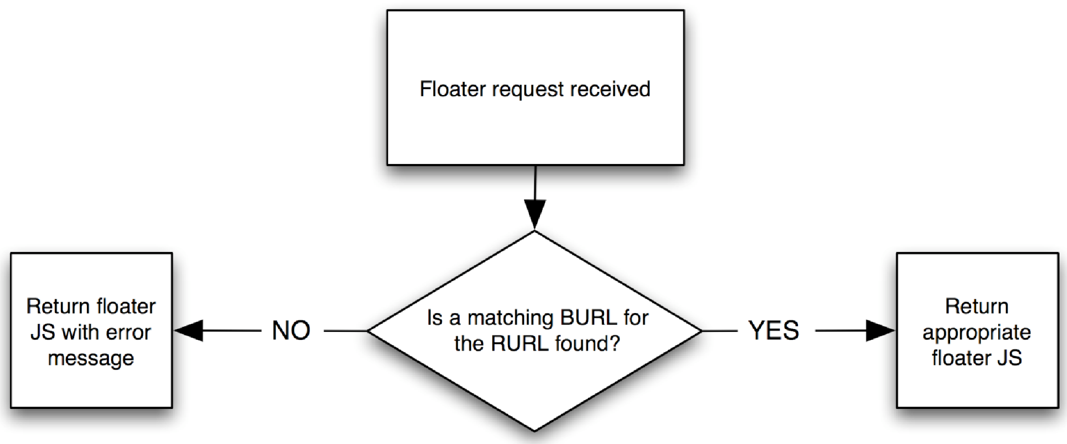

Floater request received
Is a matching BURL for the RURL found?
NO
YES
Return floater JS with error message
Return appropriate floater JS

Figure 16 – Floater process flow

Once the floater code is returned to the browser, it combines it with the content of the page. JavaScript is able to dynamically write to the XHTML source of the document. This written data acts as if it was written normally (the result of Examples 6 and 7 are identical to the user). The combination of PHP processing with JavaScript allows the floater to retrieve and display data from the database regardless of the platform of the tracked system.

```
<script language="JavaScript">
document.write('<p>Hello world!</p>');
</script>
```

Example 6 – JavaScript document writing

```
<p>Hello world!</p>
```

Example 7 – Basic HTML tagging

The floater JavaScript creates a floating layer on the document. This is achieved by placing the floater content inside a div-element. This element is assigned with a position property with the value absolute. Absolute positioning allows the placing of the div-element anywhere in the browser view port. The positioning is set by defining how many pixels from the top left corner of the view port the element should be located.

The files used by the floater are placed in a directory (public/) under the Marmalade root directory. This directory should be publicly accessible. The file structure is shown in APPENDIX F.

Setting the visual properties of the floater

The formatting of the floater is done using Cascading Style Sheets, a technology for specifying the visual properties of elements in XHTML. Today it is the standard way to set properties such as font size, font face and color in XHTML documents. A document can have references to multiple CSS files. An example of the importing tag is shown in Example 8.

```
<style type="text/css" media="screen">
@import "styles/screen.css";
</style>
```

Example 8 – XHTML tag to import an external CSS-file

In its simplest form CSS styles are set by tag, for example the td tag (table data). Example 9 demonstrates the syntax for setting a property to an element.

```
td {
background-color:green;
}
```

Example 9 – Basic CSS definition for setting table data background color

This definition sets all table data elements to have a green background, unless otherwise specified. The layout of the floater should not be affected by the CSS file of the tracked system so the visual properties should be explicitly described. This is done by using CSS class selectors and naming them correctly (as something that would unlikely be used).

```
td.marCell {
color:#212121;
background-color:white;
text-decoration:none;
font-size:11px;
font-family:arial, sans-serif;
}
```

Example 10 – Table data element class in CSS

Example 10 shows the CSS definition for the floater table data element. This definition specifies the background color of the tag explicitly. If the class selector is used (shown in Example 11) the td definition shown in Example 9 would have no effect on the cell even if both styles were imported to the document.

```
<td class=”marCell”>Table data here!</td>
```

Example 11 – Assigning a class to a table data element

Storing the user path

The floater maintains a history of the path the user has taken. This is done by storing a cookie on the user's computer. Searchsecurity.com defines cookies as follows: "A cookie is information that a Web site puts on your hard disk so that it can remember something about you at a later time. (More technically, it is information for future use that is stored by the server on the client side of a client/server communication.)" [14]. The process of using cookies is handled by the browser and is transparent to the user.

Arrays in programming languages could be considered as an advanced form of variable. They are used to store a group of related data that can be accessed using different methods (ID numbers, for example). The floater keeps track of the five latest URLs the user has visited in the tracked system. These URLs are stored into an array with a dimension of 1 by 5 (an example shown in Table 4). This array is then stored and updated to the cookie as needed. It would also be possible to use five different cookies, one for each history data. The management of five different cookies would be more complex than managing one cookie with an array.

Table 4 - Example contents of userPath array

UserPath[0]	UserPath[1]	UserPath[2]	UserPath[3]	UserPath[4]
Index.html	Articles/ index.html	Articles/technology/ index. html	Articles/technology/ biotech. html	Undefined

Cookies are, by nature, simple and hold the following data encoded in a single string:

- Cookie data
- Expiry date
- Allowed domain
- Allowed path

Storing an array into a cookie is not a standard function provided by JavaScript. Stephen Chapman's Cookie Toolbox is a JavaScript library that has functions for storing arrays into cookies. This is achieved by parsing the array into a single string when it is saved to a cookie and parsing it back to an array when one is read. The library also has other functions that allow effortless cookie management.

Example 12 shows the JavaScript function used to update the user path cookie. PHP is used to assign each cookie a unique name so the user can work on multiple floater-enabled projects without the history cookies intervening with one another. The Cookie Toolbox does not have advanced array-handling functions, so some additional functionality had to be added. In addition, the function prevents the history being filled with a single URL when the page is refreshed by comparing the latest history URL to the current one.

```
function userPathUpdate(){
    var cookieName = 'userPathCookie<?php echo $projectId ?>';
    var userPath = init_array();
    get_array(cookieName, userPath);
    var userPath5 = userPath[5];
    if(userPath5 == window.location){
        // do nothing
    } else {
    var pos = 1;
    del_entry(cookieName, userPath, pos, expires)
    get_array(cookieName, userPath);
    var num = next_entry(userPath);
    userPath[num] = window.location;
    set_array(cookieName, userPath, expires);
    }
}
```

Example 12 – The JavaScript function used to update the user path cookie

Bug submission

Once the user has filled in the reporting form and clicked the submit button the system checks that the required fields are filled. The form has three mandatory fields:

- Description
- Name
- E-mail

If one or more of these fields are missing, the operation is aborted and the user is prompted with a JavaScript alert box. If all fields are valid, the process is taken forward. The data is sent to a PHP script as HTTP post data. A complete list of the fields sent is shown in Table 5. The script handling the submitted data simply saves it to the appropriate fields in the database and prints a confirmation report to the user.

Table 5 - HTTP post data sent by the floater

Variable	**Type**	**Description**	**Example**
Description	User input	Detailed description of problem	The third level navigation is not shown if user does not have the Macromedia Flash plug-in installed.
Category	User input	Category of the problem (content, technical, visual, other)	Technical
Priority	User input	Priority of the problem (high, medium, low)	Medium
Sender	User input	Name of the person who reported the problem	James Maximi
E-mail	User input	E-mail of the person who reported the problem	James@example.com

Variable	**Type**	**Description**	**Example**
Report	User input	Is an e-mail report sent to the reporter once the problem has been fixed?	True
Client ID	Automatic	Which client was used to report the problem	Floater
Window size	Automatic	The size of the browser view port in pixels	860 x 895
Color depth	Automatic	The color depth of the users operating system	32 bits
Page title	Automatic	Title of the document (fetched from the title tag)	News service > Articles > Technology > Bio technologies
Project ID	Automatic	The project ID number in the Marmalade system	147
User Path 1	Automatic	User path data	Index.html
User Path 2	Automatic	User path data	Articles/index.html
User Path 3	Automatic	User path data	Articles/technology/ index.html
User Path 4	Automatic	User path data	Articles/technology/ biotech.html
User Path 5	Automatic	User path data	Undefined
Name	Attachment data	Name of the attached file	Screen_capture.gif
Type	Attachment data	The file type the attached file	Image/gif
Temporary Name	Attachment data	The path to the temporary file the attachment data is saved (on the server)	/var/tmp/phpIaFLNU
Size	Attachment data	Size of the attached file	23942

Browser compatibility

The floater was tested to work on the latest versions of the four popular browsers: Microsoft Internet Explorer, Mozilla Firebird, Opera and Safari. In addition, many browsers today use a common rendering engine and thus the number of compatible browsers is higher. The latest generation Netscape browsers and Mozilla, for example, both utilize the Gecko rendering engine.

In their default configurations all of the browsers mentioned above have both JavaScript and Cookies enabled. If the user disables either one of these the floater will cease to function. Support for old browser versions such as Netscape 4.x.x was not required because development teams tend to use up-to-date browser versions.

Setup of floater using the administration interface

The administration interface can be used to set some properties of the floater. Each project has its distinctive set of values. The available properties are listed in Table 6.

Table 6 - Administration interface options for the floater

Property	**Explanation**
Visibility	Sets the visibility of the floater. Possible values are enabled and disabled.
Limit by IP	Sets whether the visibility of the floater is validated against the IP-addresses in the database. The floater does not print the reporting layer for invalid IPs. Possible values are enabled and disabled.
Valid IPs	Allows the user to add and remove valid IP-addresses. They are overlooked unless the Limit by IP option is enabled.
Align	Sets the horizontal alignment of the floater element. Possible values are left and right.
Margin	Sets the distance of the floater element from the top of the browser view port. Any reasonable numerical values are accepted.

Encountered problems

The most serious difficulty was the unreliability of receiving the referrer data from the visiting browsers. The problem arises if the user installs firewall software that alters the browsers functionality so that it no longer sends the data. Some versions of Norton Firewall enable this type of behavior. Secure Hyper Text Transfer Protocol (SHTTP) connections also prevent the sending of referrer data to insecure sites (in this case the server running the Marmalade system).

The solution for this problem is to create custom floater units with hard coded project ID numbers. This removes the dependency for the referrer data. It adds complexity to updating the floater, since altering a single file no longer affects all projects.

Getting the floater to work flawlessly on several projects at the same time was also somewhat of a challenge. Due to the browsers' caching feature the JavaScript file used to display the floater was not refreshed if the user switched from one project to another. This resulted in corrupted data as project IDs and URLs were incorrect.

The solution for the browser-caching problem was to add random parameters to the URL of the floater. The URLs "http://silvia.local/marmalade/public/index.php?cat" and "http://silvia.local/marmalade/public/index.php?dog" both point to the same file. The cat and dog parameters are not used by the script, but the browser now considers them unique.

5.4. RSS feed

The RSS implementation was done after the administration interface was completed, because the RSS system relies on it to create the right queries to the database. RSS feed URLs are identical to the administration interface URLs, except for the RSS parameter. An example URL is shown in Example 13.

```
http://silvia.local/marmalade/report.php?category=%25&status=%25
&priority=%25&keyword=james&projectId=109&rss=true
```

Example 13 - An URL pointing to an RSS feed

If the RSS parameter is set to be true, the administration interface will output an RSS feed instead of the XHTML interface. Example 14 demonstrates how the system includes the rss.php file and stops executing if the parameter is set to be true.

```
// if RSS then include rss.php and exit
if ($_GET[rss] == "true"){
    $rssLink = explode("/", "$_SERVER[REQUEST_URI]");
    $rssLink = $systemUrl.end($rssLink);
    $rssLink = htmlspecialchars($rssLink);
    include("./rss.php");
    exit;
}
```

Example 14 - RSS switch code

The RSS implementation creates an RSS feed containing multiple items, one for each matching bug. Since XHTML formatting is considered valid, it is used to layout the data in tables similar to the administration interface. The basic bug item structure in an RSS feed is shown in Example 15.

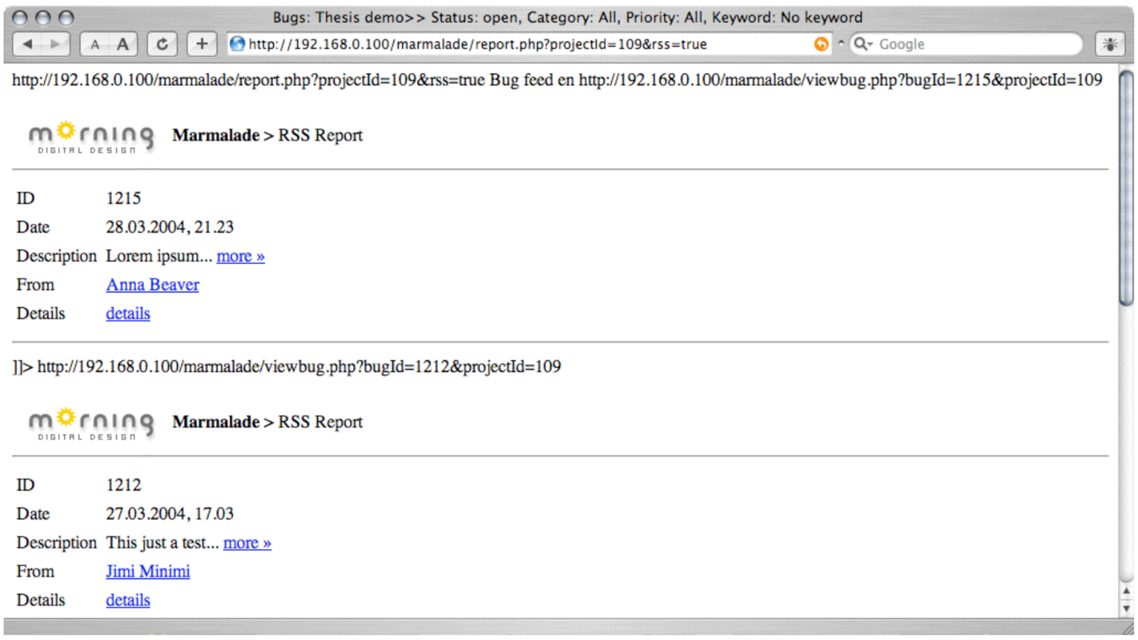
```
<item>
<title><![CDATA[Bug Entry (1208) from Jimi Minimi]]></title>
<description><![CDATA[

############################################
#                                          #
# BUG DETAILS FORMATTED WITH HTML ELEMENTS #
#                                          #
############################################


]]></description>
</item>
```

Example 15 - Marmalade 1.0 RSS item

The RSS feed can be read by browsers, but may seem corrupted to the user. An example is shown in Image 5. The real strength of RSS feeds lie in the reader applications. These applications can interpret the feed into an organized listing of items. Screen captures of suggested RSS reader applications are shown in APPENDIX C.


Bugs: Thesis demo>> Status: open, Category: All, Priority: All, Keyword: No keyword
http://192.168.0.100/marmalade/report.php?projectId=109&rss=true
Google

http://192.168.0.100/marmalade/report.php?projectId=109&rss=true Bug feed en http://192.168.0.100/marmalade/viewbug.php?bugId=1215&projectId=109

Marmalade > RSS Report

ID	1215
Date	28.03.2004, 21.23
Description	Lorem ipsum... more »
From	Anna Beaver
Details	details

]]> http://192.168.0.100/marmalade/viewbug.php?bugId=1212&projectId=109

Marmalade > RSS Report

ID	1212
Date	27.03.2004, 17.03
Description	This just a test... more »
From	Jimi Minimi
Details	details

Image 5 - RSS feed shown in the Safari browser

Problems encountered

Due to its simplicity, the RSS feed was graceful to implement. However, problems did arise with the suitability of reader applications. Most reader applications use a caching method to store feed items. For the Marmalade system, this was a problem since the aim was to provide the users with a live view to the database.

For example, a user wishes to receive a feed with bugs marked as open and categorized as content. Caching readers function correctly as long as no items are removed from the feed. In this case removing an item would constitute setting a bug's status as closed. Caching readers will not recognize the removal of this item and continue to store it in the cache. To the user of the system it appears as if no bugs are being corrected.

The solution for this problem was to find reader applications that are able to function without caching the headlines. The recommended RSS reader applications for the Marmalade 1.0 system are Wildgrape NewsDesk for Microsoft Windows and Ranchero Software NetNewsWire Lite for Macintosh environments. Screen captures of the applications are shown in APPENDIX C.

5.5. Platform transition

The target platform must meet the following requirements:

- Apache 1.3.x installed
- MySQL 4.x.x installed
- PHP 4.x.x (with MySQL & LDAP support compiled) installed
- Sendmail installed

The platform transition process has three steps:

- Move files
- Import database structure
- Modify configuration file

The first step is to move the files to the remote host. The command is shown in Example 16.

```
scp -r marmalade janit@madeleine.local:~/public_html/
```

Example 16 – Secure copy command

Once the files are moved, the database is created and the schema is loaded. The schema is included in the basic file structure (marmalade.sql). The commands shown in Example 17 are executed on the production server.

```
mysql> create database marmalade;
Query OK, 1 row affected (0.00 sec)

mysql> exit;
Bye
madeleine:~ janit$ mysql -u root -p marmalade <
marmalade/marmalade.sql
Enter password:
madeleine:~ janit$
```

Example 17 – Database create and schema load commands

Once the database is working (set user permissions if needed), the configuration file (shown in Example 18) is opened with a text editor such as EMACS or Pico.

```
$today = date("d.m.Y, G.i");
$systemUrl = "http://madeleine.local/marmalade/";
$dbHost = 'localhost';
$dbUser = 'root';
$dbPass = 'password';
$dbName = 'marmalade';
```

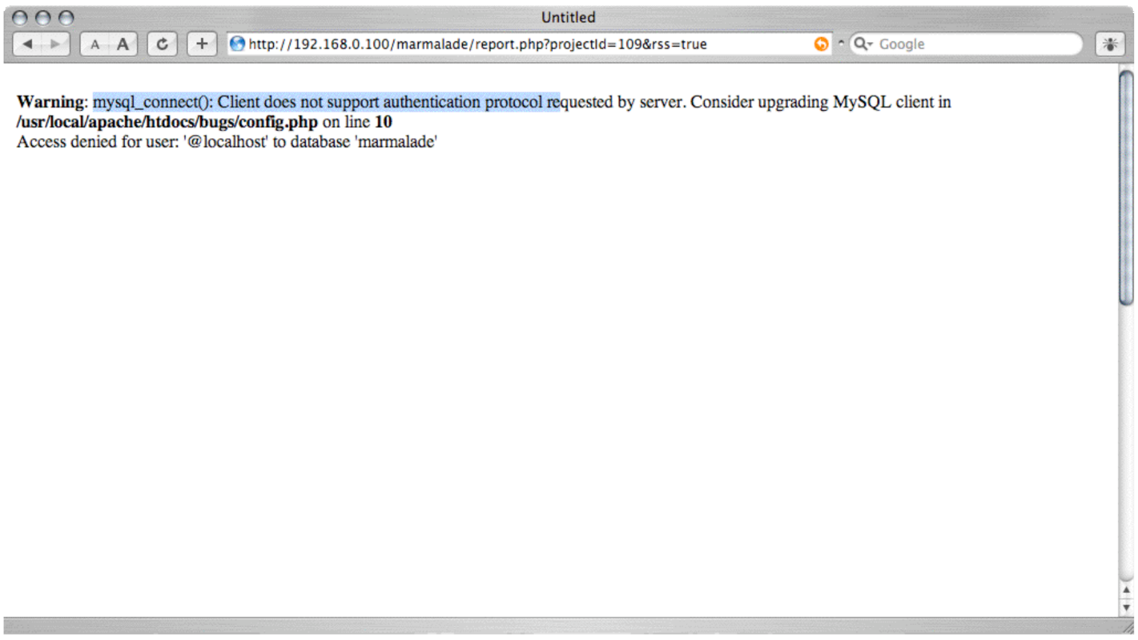
Example 18 – main configuration file (config.php)

Once these steps are completed, the system is ready. Test by opening the system URL.

Encountered problems

Some minor problems due to differences in environment settings were encountered:

- The PHP configuration required that all files be addressed with "./"
- Missing LDAP functions (not compiled to PHP)
- MySQL password format difference (screen capture shown in Image 6)


Untitled
http://192.168.0.100/marmalade/report.php?projectId=109&rss=true
Google

Warning: mysql_connect(): Client does not support authentication protocol requested by server. Consider upgrading MySQL client in **/usr/local/apache/htdocs/bugs/config.php** on line **10**
Access denied for user: '@localhost' to database 'marmalade'

Image 6 – MySQL password format incompatibility error shown by the application

6. Floater deployment

The deployment of the floater client involves adding a tag to the XHTML of the tracked system. The required tag is a script tag, which refers to an external source. The system relies on the referrer data given by the browser, so the JavaScript code cannot be placed inside the script tags. The placement of the tag is not crucial, but it would be good practice to always place it to the same place, for example before the closing body tag.

To avoid browser caching problems with multiple projects a random parameter should be added by appending the following characters to the source parameter: "?rand=XXXXXX" (the Xs represent random characters). The random parameter should be same for each page of the project. If a random number is generated, the floater code is loaded each time. This results in increased server load.

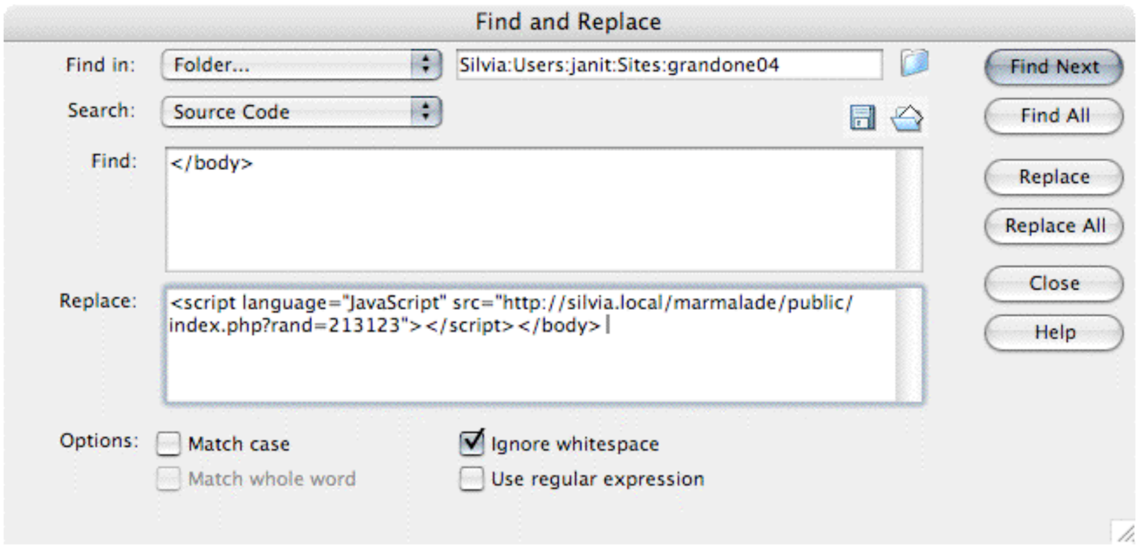
An example of the required script tag and its placement in the XHTML file is shown in Example 19.

```
<script language="JavaScript"
src="http://silvia.local/marmalade/public/index.php?rand=880734116">
</script>
</body>
```

Example 19 – Script tag for loading the floater

The code should be on each individual XHTML-page the browser receives. For template-based systems, it should be added to each template or possibly to the commonly included footer file.

For tracking static XHTML-files, the process requires additional tools or tedious handwork. It is possible to make the changes by hand to each file, but it is ideal to use a search and replace function of a text editor. This function searches a mass of files for a certain pattern and replaces it with another. This feature is common in popular editor tools (such as Macromedia Homesite and Dreamweaver) and makes it effortless to make the required changes. The search and replace interface of Macromedia Dreamweaver is shown in Image 7.



Find in: Folder... Silvia:Users:janit:Sites:grandone04

Search: Source Code

Find:

```
</body>
```

Replace:

```
<script language="JavaScript" src="http://silvia.local/marmalade/public/
index.php?rand=213123"></script></body>
```

Options: ☐ Match case ☑ Ignore whitespace ☐ Match whole word ☐ Use regular expression

Image 7 - Macromedia Dreamweaver search and replace interface

It is recommended that the script tags be removed from the system when it enters the production stage. In theory with the floater visibility property set to hidden the floater should not be displayed to the users. In practice, the XHTML can be examined by the users of a web site (or a web application) and this could present a security risk. In addition not removing the tag would create unnecessary load for the server running the Marmalade 1.0 system.

7. Conclusions

The purpose of the project was to create a tool for reporting and keeping track of bugs. In technical terms, the project was successful as the final product fulfills the goals set during the requirements phase. The development process followed the original plan and was in all quite fluent. The reason for this could be the fact that the implementation was done independently, unaffected by other people's lack of time, interest or motivation.

Knowledge of the theory of common bug tracking practices and methods were gained during the requirements and planning stages. The implementation phase was technically challenging and further experience of web application development was acquired. Especially the implementation of the floater by combining server-side and client-side scripting proved to be challenging. It also gave new insight into browser behavior, as well as their possibilities and potential problems.

In its current version, the system is limited to tracking pages within XHTML based applications. With the rising number of Macromedia Flash content created at Morning, a floater unit similar to the JavaScript version could be built using Flash's ActionScript.

In April 2004 the Marmalade 1.0 system has been used on two projects within the company. One of them is an ongoing internal development project. The other was a client project, which was completed during an intensive two-week period. In both cases, the system has worked well in technical terms; there were few problems in reporting bugs or administering them.

The real issues lay in the fact that the tool has not yet been adopted as a part of the development process. Before the system can be successful, it needs to be a standard practice in projects. In addition, it is important to train the reporters how to clarify the problem at hand: "A report that vaguely states that the navigation is confusing will be worthless. Instead, it needs to state specifically how the navigation might be improved. Then the developers need to return to the code and fix the problem." [2, p. 407]

References

[1] Matthew A. Telles, Yuan Hsieh. The Science of Debugging. 2001, The Coriolis Group.

[2] Darren Gergle, Tom Brinck, Scott D. Wood. Usability for the Web: Designing web sites that work. 2002, Academic Press.

[3] Morning Digital Design Oy. 2004, WWW-document. http://www.morning.fi/en/

[4] Steve Maguire. Writing Solid Code. 1993, Microsoft Press.

[5] Jakob Nielsen. Designing Web Usability: The Practice of Simplicity. 2000, New Riders Publishing.

[6] Joel Spolsky. Painless Bug Tracking. 2000, WWW-document. http://www.joelonsoftware.com/articles/fog0000000029.html

[7] Free Software Foundation. Free Software Directory. WWW-document. http://www.gnu.org/directory/devel/debug/Bug_tracking_systems/

[8] Free Software Foundation. General Public License. 1991, WWW-document. http://www.gnu.org/licenses/gpl.html

[9] The Mozilla Organization. WWW-document. http://www.mozilla.org/download.html

[10] The Internet Engineering Task Force. Overview of the IETF, WWW-document. http://www.ietf.org/overview.html

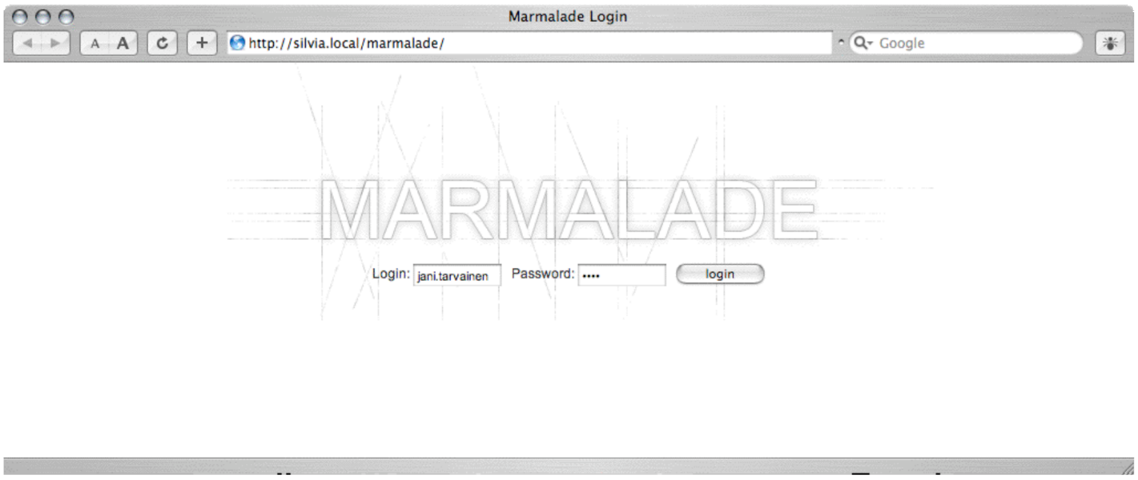
[11] Mark Pilgrim. What is RSS. 2002, WWW-document.
http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html

[12] Mark Nottingham. RSS tutorial for Content Publishers and Webmasters. 2002, WWW-document.
http://www.mnot.net/rss/tutorial/

[13] Thomas J. Shelford, Gregory A. Remillard, Michael Smith. Real Web Project management. 2003, Addison-Wesley.

[14] TechTarget. Cookie definition. WWW-document.
http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci211838,00.html

APPENDIX A: Administration interface screen captures

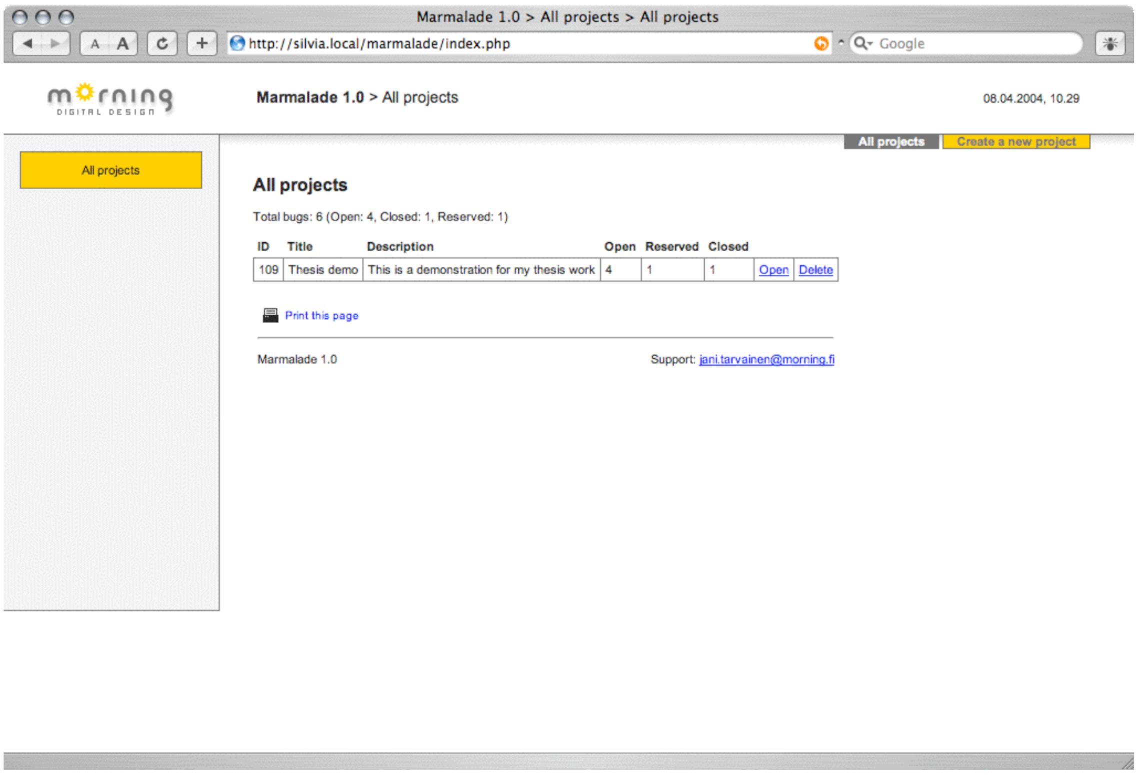

Marmalade Login

http://silvia.local/marmalade/

MARMALADE

Login: jani.tarvainen Password: •••• login

Image 8 - Administration interface (login screen)


Marmalade 1.0 > All projects > All projects

http://silvia.local/marmalade/index.php

morning DIGITAL DESIGN

Marmalade 1.0 > All projects

08.04.2004, 10.29

All projects | Create a new project

All projects

All projects

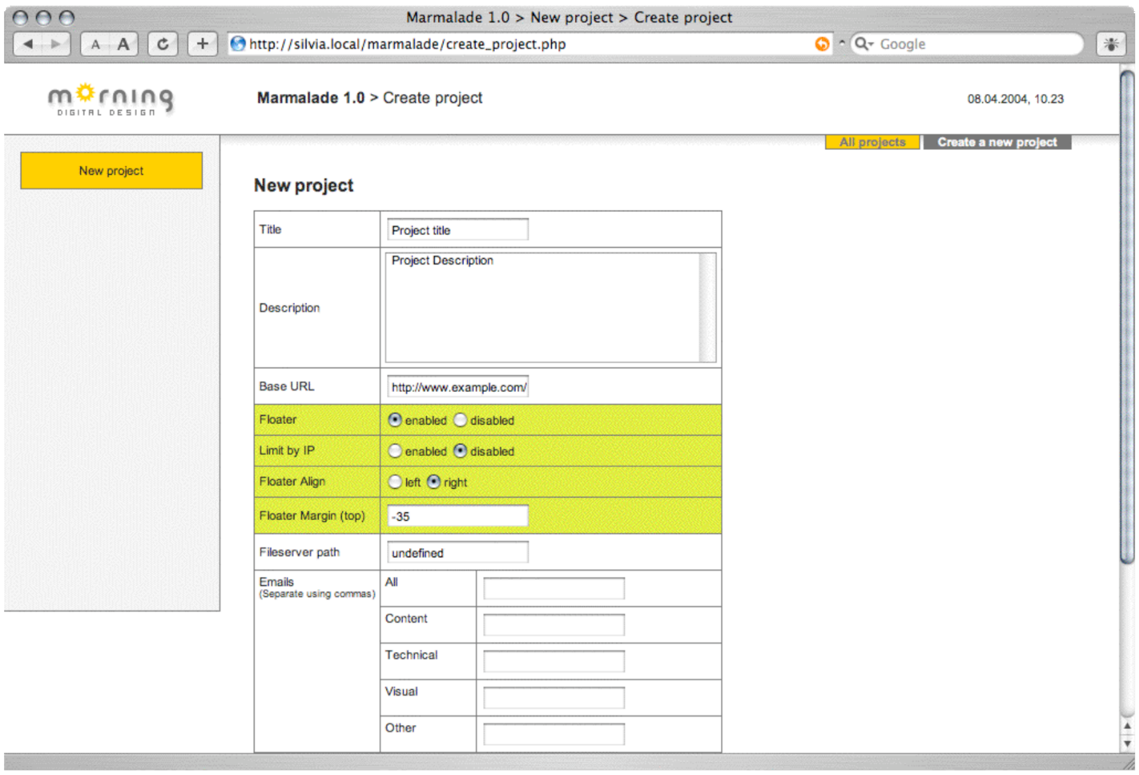
Total bugs: 6 (Open: 4, Closed: 1, Reserved: 1)

ID	Title	Description	Open	Reserved	Closed		
109	Thesis demo	This is a demonstration for my thesis work	4	1	1	Open	Delete

Print this page

Marmalade 1.0 Support: jani.tarvainen@morning.fi

Image 9 - Administration interface (project listing)



Marmalade 1.0 > New project > Create project
http://silvia.local/marmalade/create_project.php
Google
morning DIGITAL DESIGN
All projects
Create a new project
New project

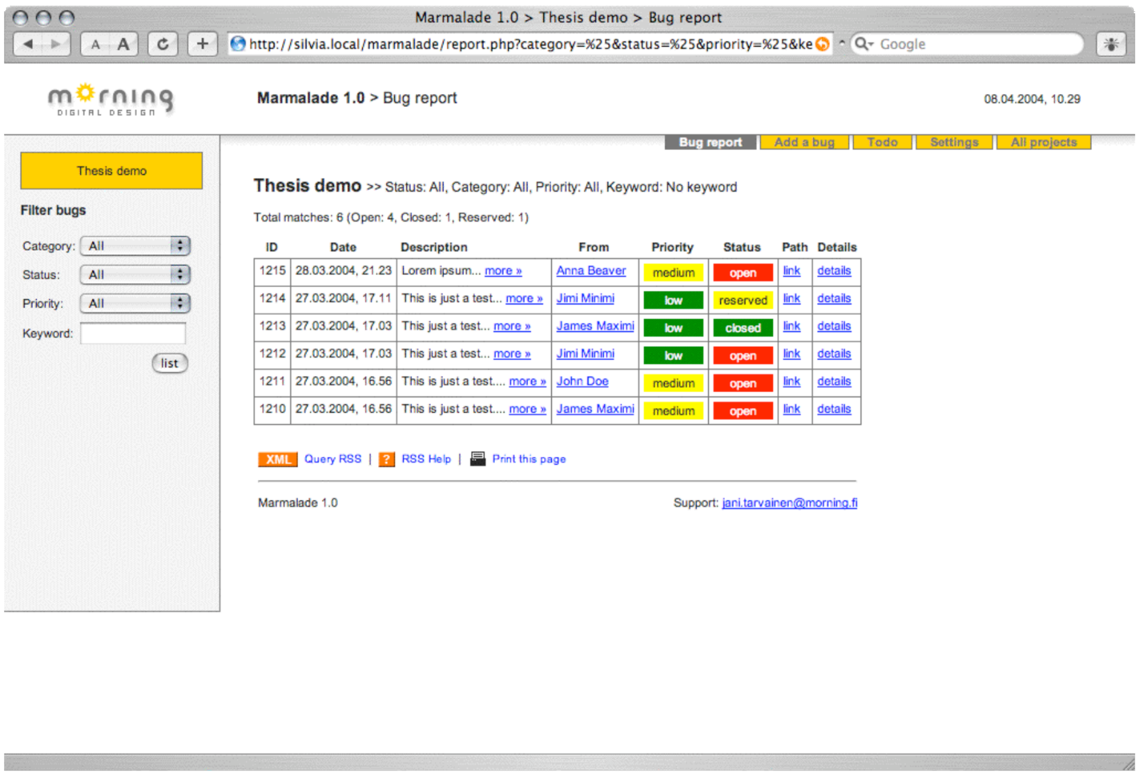
Marmalade 1.0 > Create project

08.04.2004, 10.23

New project

Title	Project title
Description	Project Description
Base URL	http://www.example.com/
Floater	[✓ enabled] disabled
Limit by IP	enabled [✓ disabled]
Floater Align	left [✓ right]
Floater Margin (top)	-35
Fileserver path	undefined
Emails (Separate using commas)	All
	Content
	Technical
	Visual
	Other

Image 10 - Administration interface (create project)



Marmalade 1.0 > Bug report

08.04.2004, 10.29

Filter bugs

Category: All
Status: All
Priority: All
Keyword:

Thesis demo >> Status: All, Category: All, Priority: All, Keyword: No keyword

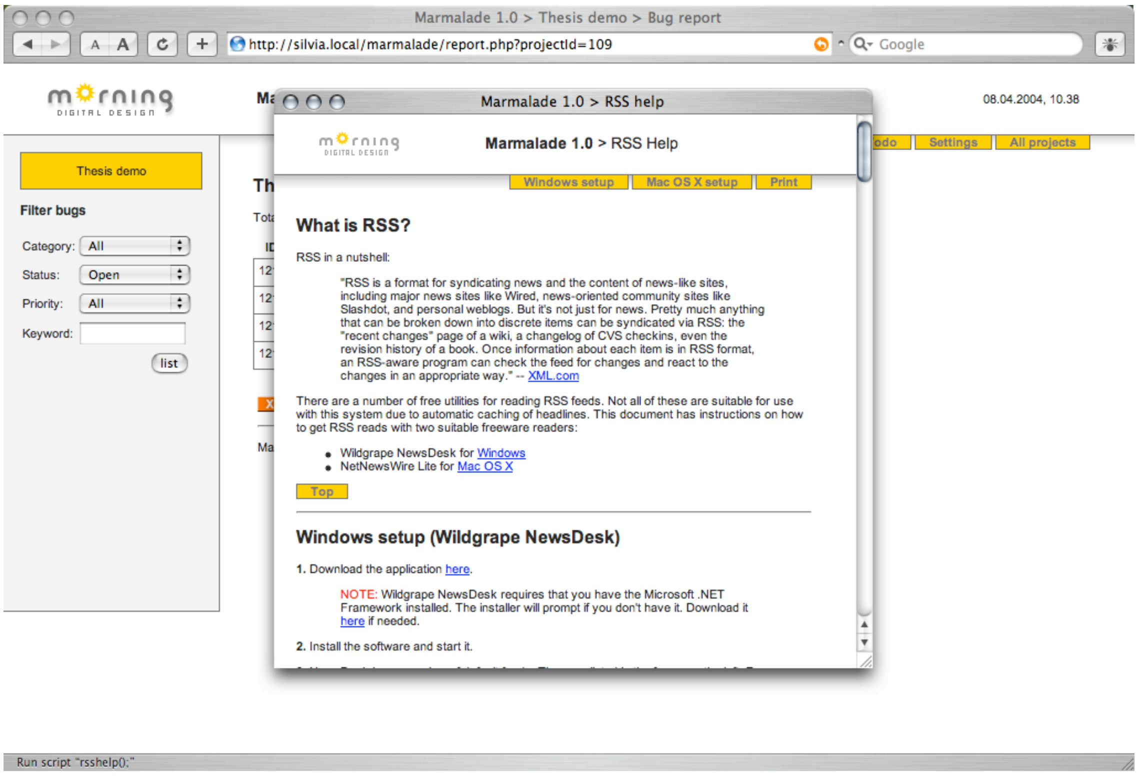
Total matches: 6 (Open: 4, Closed: 1, Reserved: 1)

ID	Date	Description	From	Priority	Status	Path	Details
1215	28.03.2004, 21.23	Lorem ipsum... more »	Anna Beaver	medium	open	link	details
1214	27.03.2004, 17.11	This is just a test... more »	Jimi Minimi	low	reserved	link	details
1213	27.03.2004, 17.03	This just a test... more »	James Maximi	low	closed	link	details
1212	27.03.2004, 17.03	This just a test... more »	Jimi Minimi	low	open	link	details
1211	27.03.2004, 16.56	This is just a test.... more »	John Doe	medium	open	link	details
1210	27.03.2004, 16.56	This is just a test.... more »	James Maximi	medium	open	link	details

Marmalade 1.0

Support: jani.tarvainen@morning.fi

Image 11 - Administration interface (bug listing)



Marmalade 1.0 > RSS Help

What is RSS?

RSS in a nutshell:

> "RSS is a format for syndicating news and the content of news-like sites, including major news sites like Wired, news-oriented community sites like Slashdot, and personal weblogs. But it's not just for news. Pretty much anything that can be broken down into discrete items can be syndicated via RSS: the "recent changes" page of a wiki, a changelog of CVS checkins, even the revision history of a book. Once information about each item is in RSS format, an RSS-aware program can check the feed for changes and react to the changes in an appropriate way." -- XML.com

There are a number of free utilities for reading RSS feeds. Not all of these are suitable for use with this system due to automatic caching of headlines. This document has instructions on how to get RSS reads with two suitable freeware readers:

- Wildgrape NewsDesk for Windows
- NetNewsWire Lite for Mac OS X

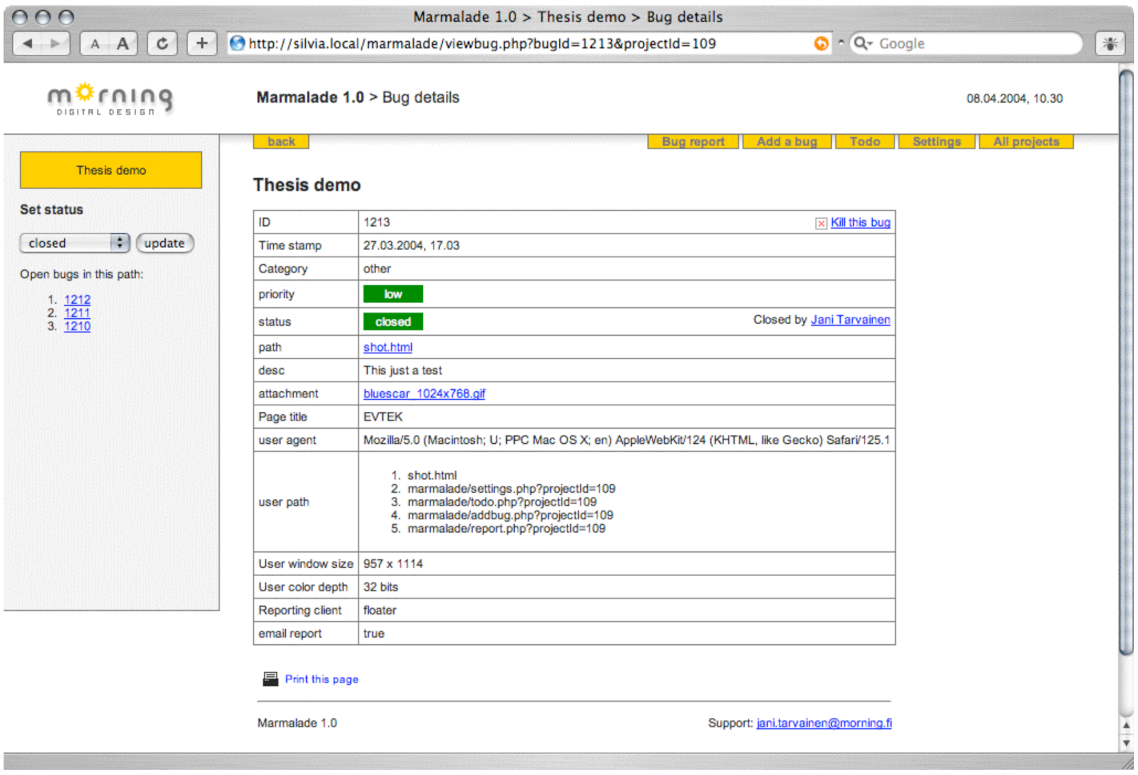
Windows setup (Wildgrape NewsDesk)

1. Download the application here.

> NOTE: Wildgrape NewsDesk requires that you have the Microsoft .NET Framework installed. The installer will prompt if you don't have it. Download it here if needed.

2. Install the software and start it.

Image 12 - Administration interface (RSS help pop-up)



Marmalade 1.0 > Bug details

Open bugs in this path:

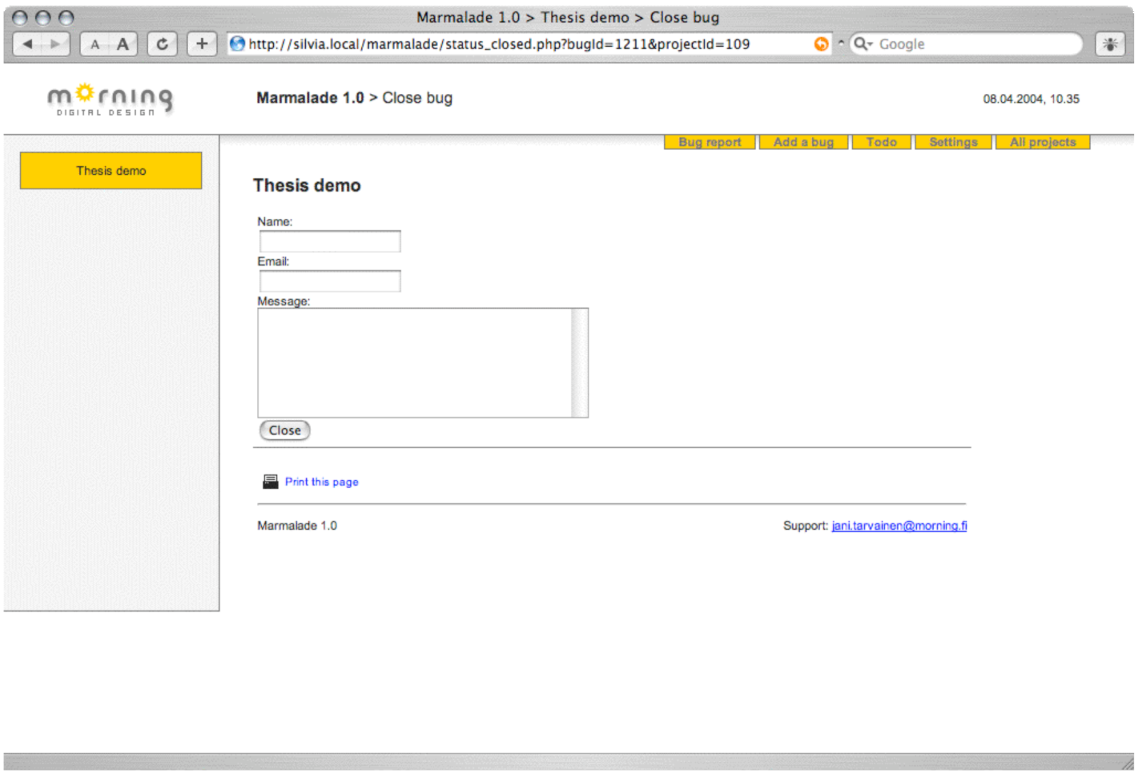
1. 1212
2. 1211
3. 1210

Thesis demo

ID	1213 Kill this bug
Time stamp	27.03.2004, 17.03
Category	other
priority	low
status	closed Closed by Jani Tarvainen
path	shot.html
desc	This just a test
attachment	bluescar_1024x768.gif
Page title	EVTEK
user agent	Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en) AppleWebKit/124 (KHTML, like Gecko) Safari/125.1
user path	1. shot.html 2. marmalade/settings.php?projectId=109 3. marmalade/todo.php?projectId=109 4. marmalade/addbug.php?projectId=109 5. marmalade/report.php?projectId=109
User window size	957 x 1114
User color depth	32 bits
Reporting client	floater
email report	true

Marmalade 1.0 Support: jani.tarvainen@morning.fi

Image 13 - Administration interface (bug details)



Marmalade 1.0 > Thesis demo > Close bug

http://silvia.local/marmalade/status_closed.php?bugId=1211&projectId=109

Marmalade 1.0 > Close bug

08.04.2004, 10.35

Bug report | Add a bug | Todo | Settings | All projects

Thesis demo

Thesis demo

Name:

Email:

Message:

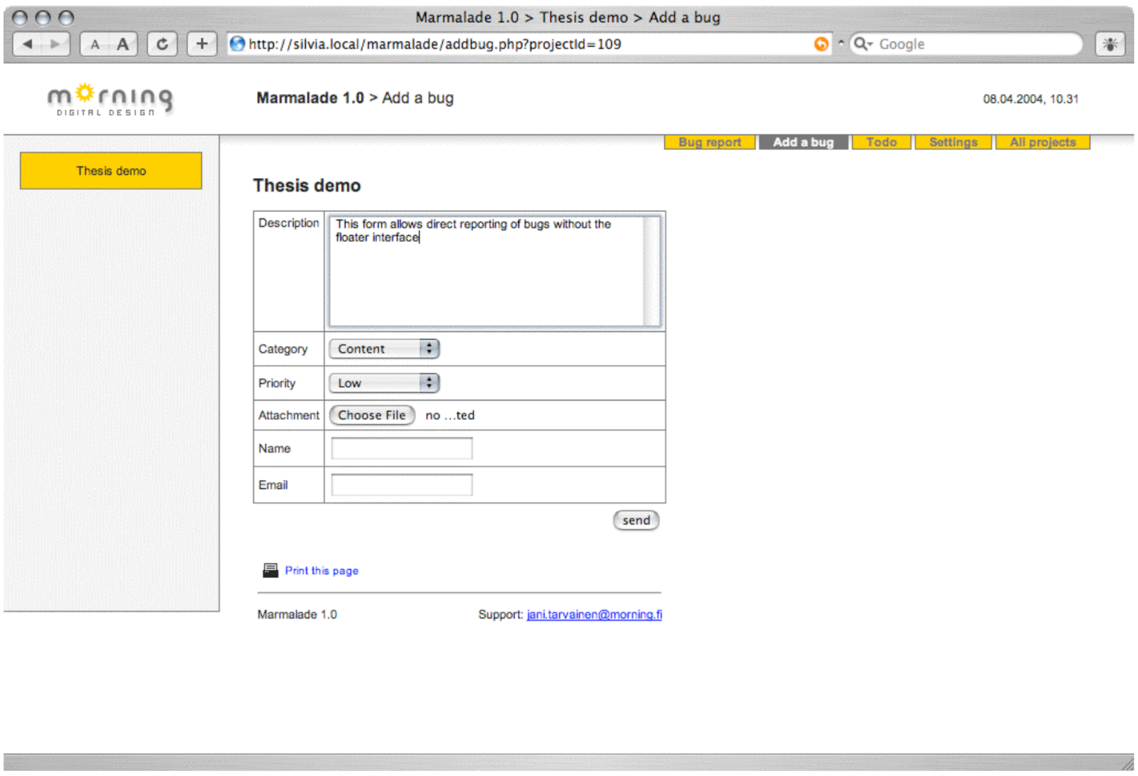
Close

Print this page

Marmalade 1.0

Support: jani.tarvainen@morning.fi

Image 14 - Administration interface (bug closing form)



Marmalade 1.0 > Thesis demo > Add a bug

http://silvia.local/marmalade/addbug.php?projectId=109

Marmalade 1.0 > Add a bug

08.04.2004, 10.31

Bug report | Add a bug | Todo | Settings | All projects

Thesis demo

Thesis demo

Description	This form allows direct reporting of bugs without the floater interface
Category	Content
Priority	Low
Attachment	Choose File no …ted
Name	
Email	

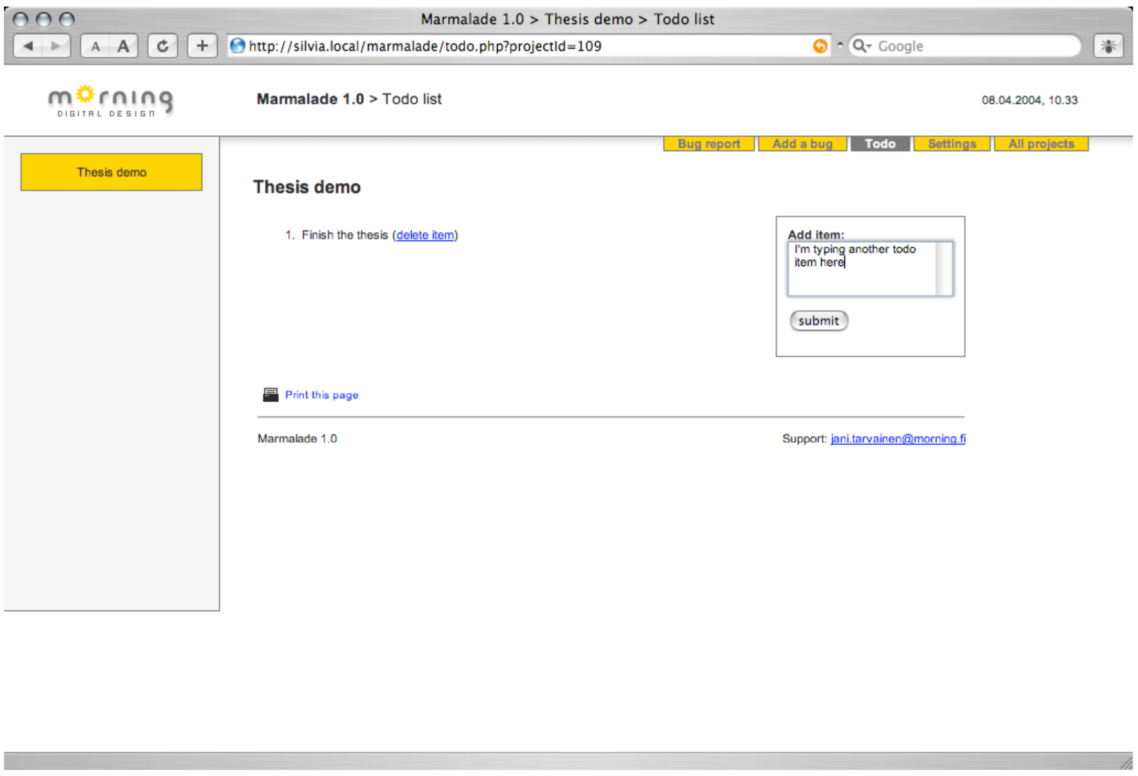
send

Print this page

Marmalade 1.0

Support: jani.tarvainen@morning.fi

Image 15 - Administration interface (bug reporting form)



http://silvia.local/marmalade/todo.php?projectId=109

Marmalade 1.0 > Todo list — 08.04.2004, 10.33

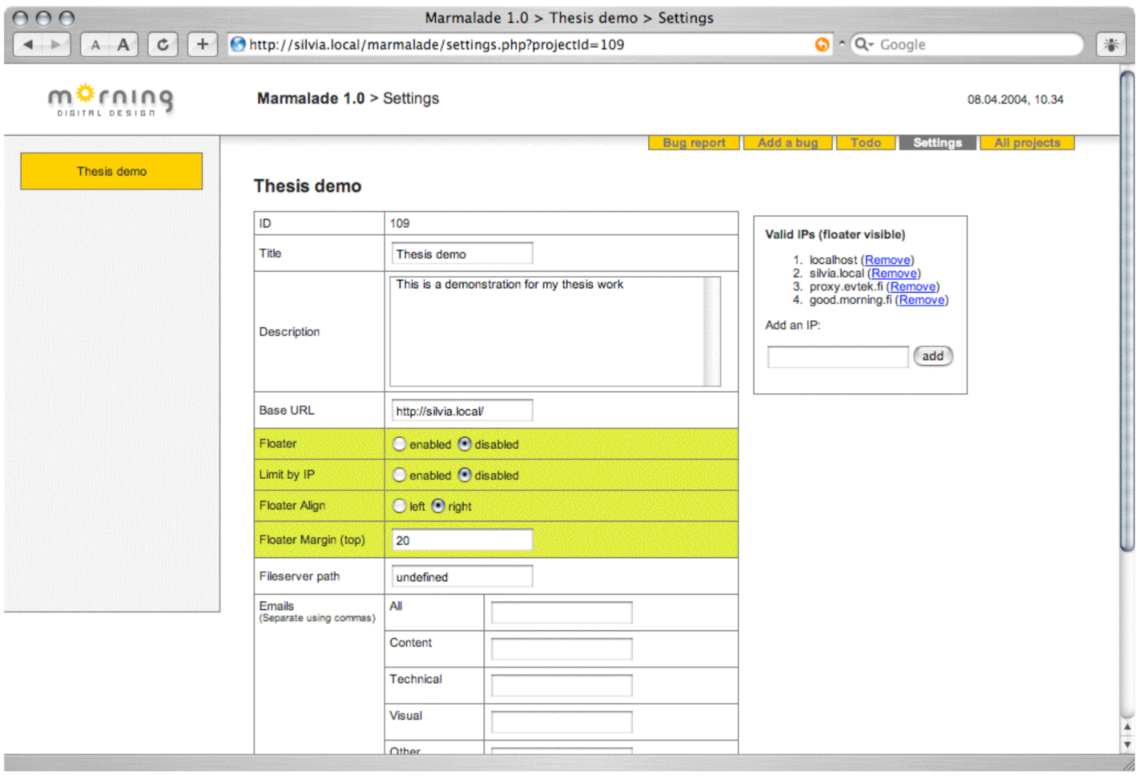
Thesis demo

1. Finish the thesis (delete item)

Add item:
I'm typing another todo item here

Marmalade 1.0 — Support: jani.tarvainen@morning.fi

Image 16 - Administration interface (todo listing)



http://silvia.local/marmalade/settings.php?projectId=109

Marmalade 1.0 > Settings — 08.04.2004, 10.34

Thesis demo

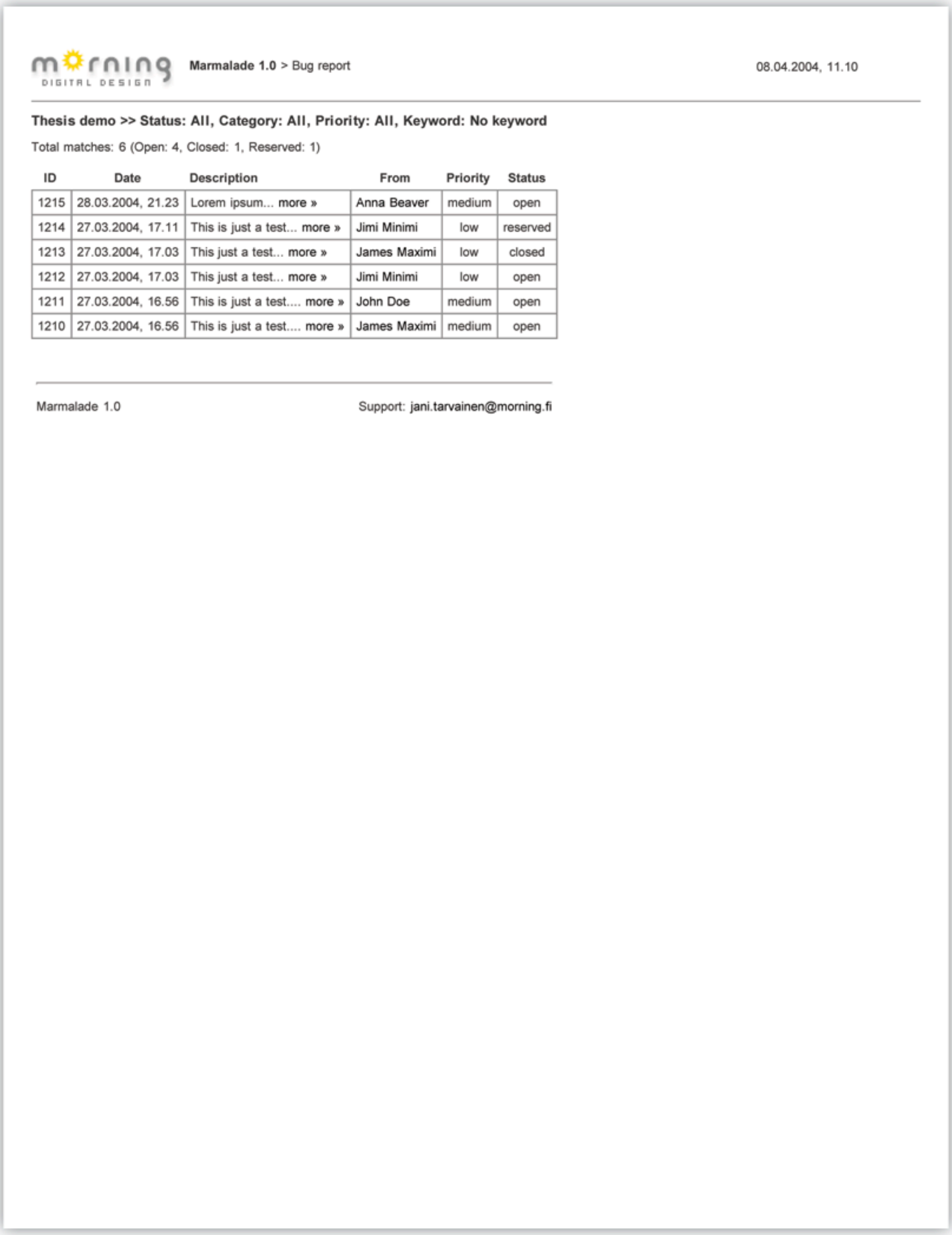
ID	109	
Title	Thesis demo	
Description	This is a demonstration for my thesis work	
Base URL	http://silvia.local/	
Floater	enabled [✓ disabled]	
Limit by IP	enabled [✓ disabled]	
Floater Align	left [✓ right]	
Floater Margin (top)	20	
Fileserver path	undefined	
Emails (Separate using commas)	All	
	Content	
	Technical	
	Visual	
	Other	

Valid IPs (floater visible)

1. localhost (Remove)
2. silvia.local (Remove)
3. proxy.evtek.fi (Remove)
4. good.morning.fi (Remove)

Add an IP:

Image 17 - Administration interface (project settings)



Marmalade 1.0 > Bug report

08.04.2004, 11.10

Thesis demo >> Status: All, Category: All, Priority: All, Keyword: No keyword

Total matches: 6 (Open: 4, Closed: 1, Reserved: 1)

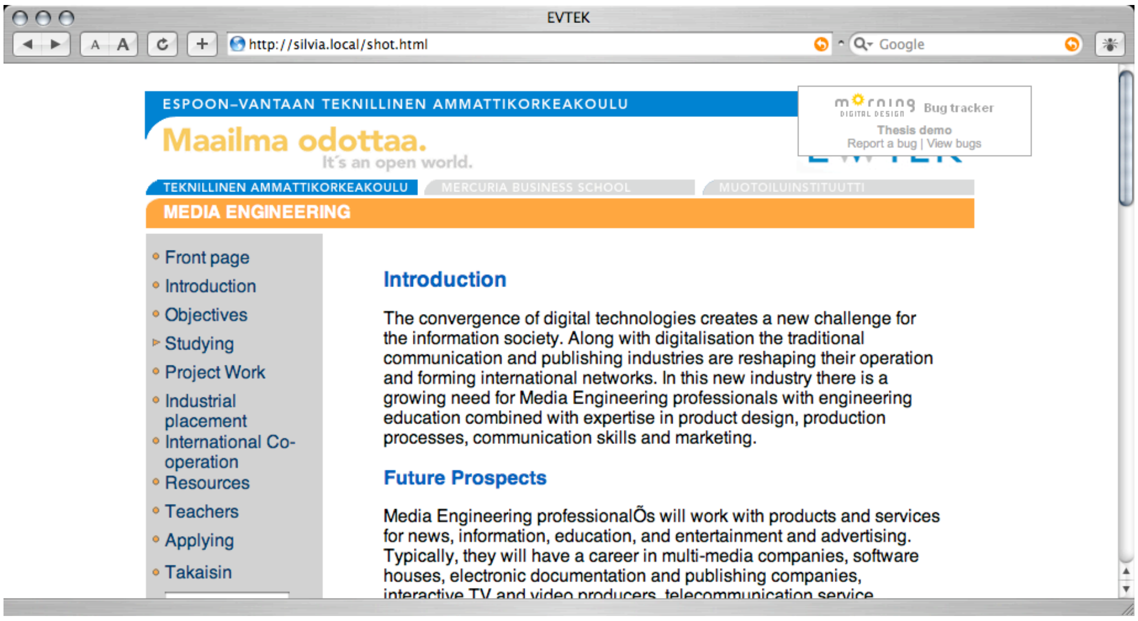
ID	Date	Description	From	Priority	Status
1215	28.03.2004, 21.23	Lorem ipsum... more »	Anna Beaver	medium	open
1214	27.03.2004, 17.11	This is just a test... more »	Jimi Minimi	low	reserved
1213	27.03.2004, 17.03	This just a test... more »	James Maximi	low	closed
1212	27.03.2004, 17.03	This just a test... more »	Jimi Minimi	low	open
1211	27.03.2004, 16.56	This is just a test.... more »	John Doe	medium	open
1210	27.03.2004, 16.56	This is just a test.... more »	James Maximi	medium	open

Marmalade 1.0

Support: jani.tarvainen@morning.fi

Image 18 - Administration interface (printed report)

APPENDIX B: Floater interface screen captures



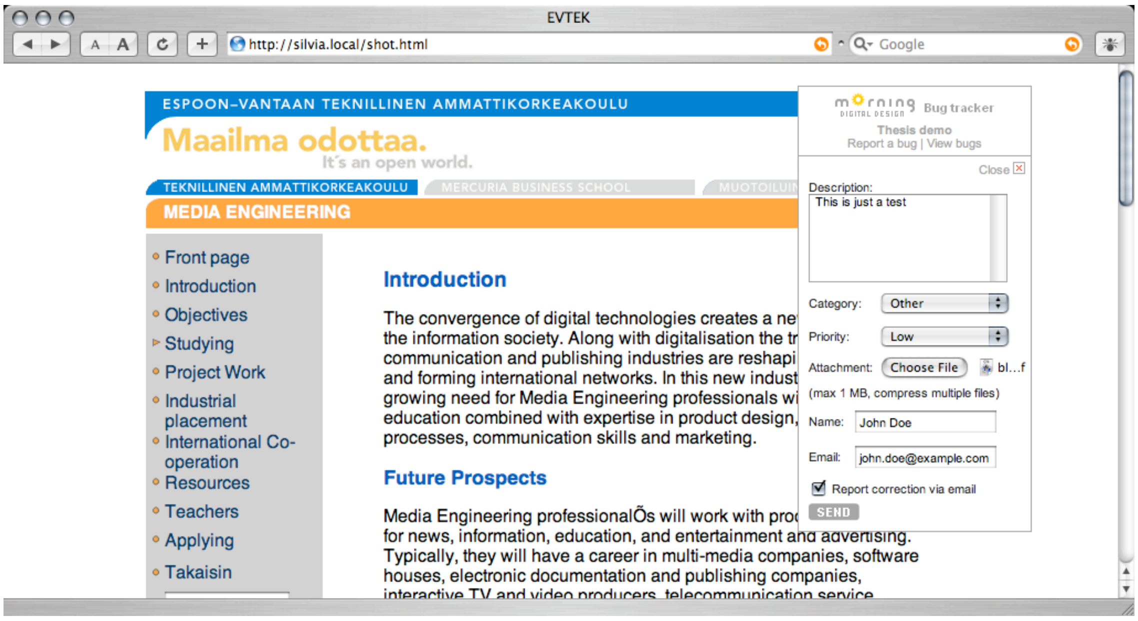
Introduction

The convergence of digital technologies creates a new challenge for the information society. Along with digitalisation the traditional communication and publishing industries are reshaping their operation and forming international networks. In this new industry there is a growing need for Media Engineering professionals with engineering education combined with expertise in product design, production processes, communication skills and marketing.

Future Prospects

Media Engineering professionalÕs will work with products and services for news, information, education, and entertainment and advertising. Typically, they will have a career in multi-media companies, software houses, electronic documentation and publishing companies, interactive TV and video producers, telecommunication service

Image 19 - Floater interface (report form closed)



Description: This is just a test

Category: Other

Priority: Low

Attachment: bl…f

(max 1 MB, compress multiple files)

Name: John Doe

Email: john.doe@example.com

[✓] Report correction via email

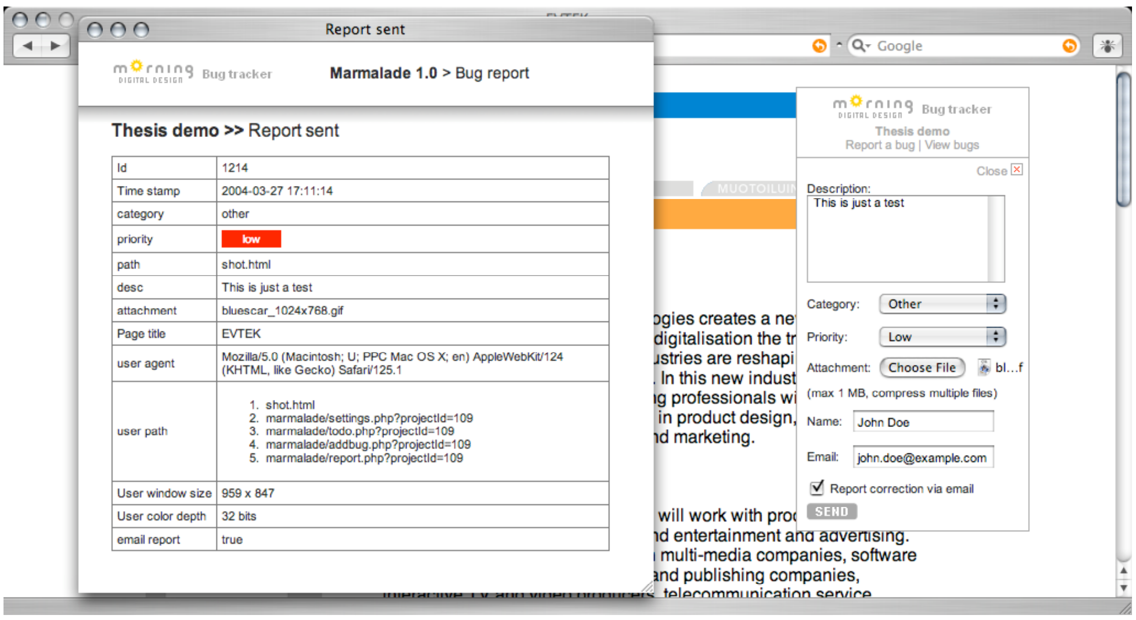
Introduction

The convergence of digital technologies creates a ne
the information society. Along with digitalisation the tr
communication and publishing industries are reshapi
and forming international networks. In this new indust
growing need for Media Engineering professionals wi
education combined with expertise in product design,
processes, communication skills and marketing.

Future Prospects

Media Engineering professionalÕs will work with proc
for news, information, education, and entertainment and advertising.
Typically, they will have a career in multi-media companies, software
houses, electronic documentation and publishing companies,
interactive TV and video producers, telecommunication service

Image 20 – Floater interface (report form open)


Report sent

Marmalade 1.0 > Bug report

Thesis demo >> Report sent

Id	1214
Time stamp	2004-03-27 17:11:14
category	other
priority	low
path	shot.html
desc	This is just a test
attachment	bluescar_1024x768.gif
Page title	EVTEK
user agent	Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en) AppleWebKit/124 (KHTML, like Gecko) Safari/125.1
user path	1. shot.html 2. marmalade/settings.php?projectId=109 3. marmalade/todo.php?projectId=109 4. marmalade/addbug.php?projectId=109 5. marmalade/report.php?projectId=109
User window size	959 x 847
User color depth	32 bits
email report	true

Thesis demo
Report a bug | View bugs

Description: This is just a test

Category: Other

Priority: Low

Attachment: bl…f

(max 1 MB, compress multiple files)

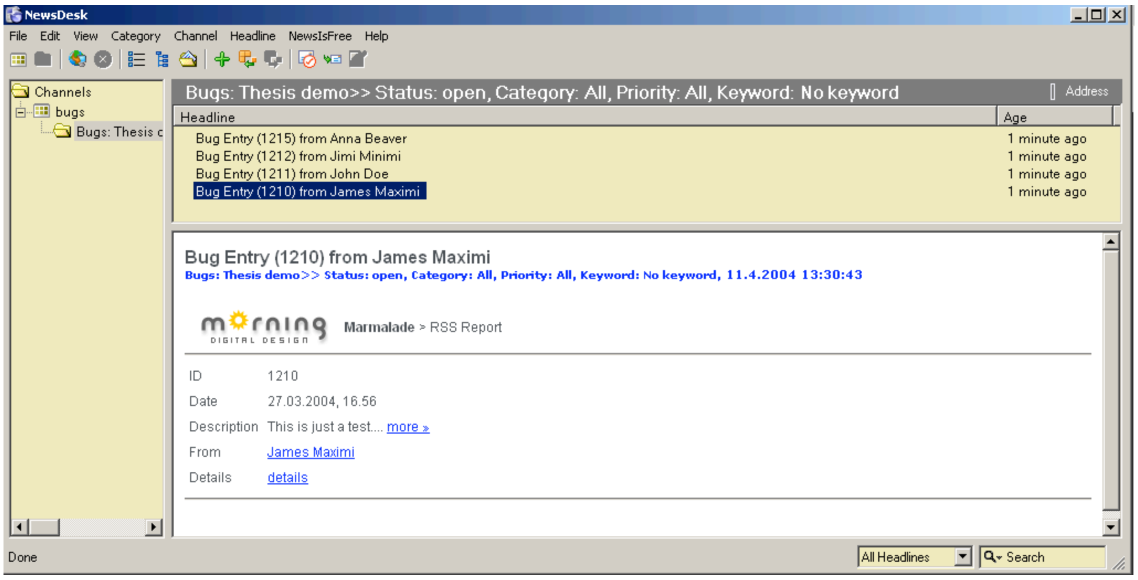
Name: John Doe

Email: john.doe@example.com

✓ Report correction via email

Image 21 - Floater interface (successful report)

APPENDIX C: RSS reader screen captures



Bugs: Thesis demo>> Status: open, Category: All, Priority: All, Keyword: No keyword

Headline	Age
Bug Entry (1215) from Anna Beaver	1 minute ago
Bug Entry (1212) from Jimi Minimi	1 minute ago
Bug Entry (1211) from John Doe	1 minute ago
Bug Entry (1210) from James Maximi	1 minute ago

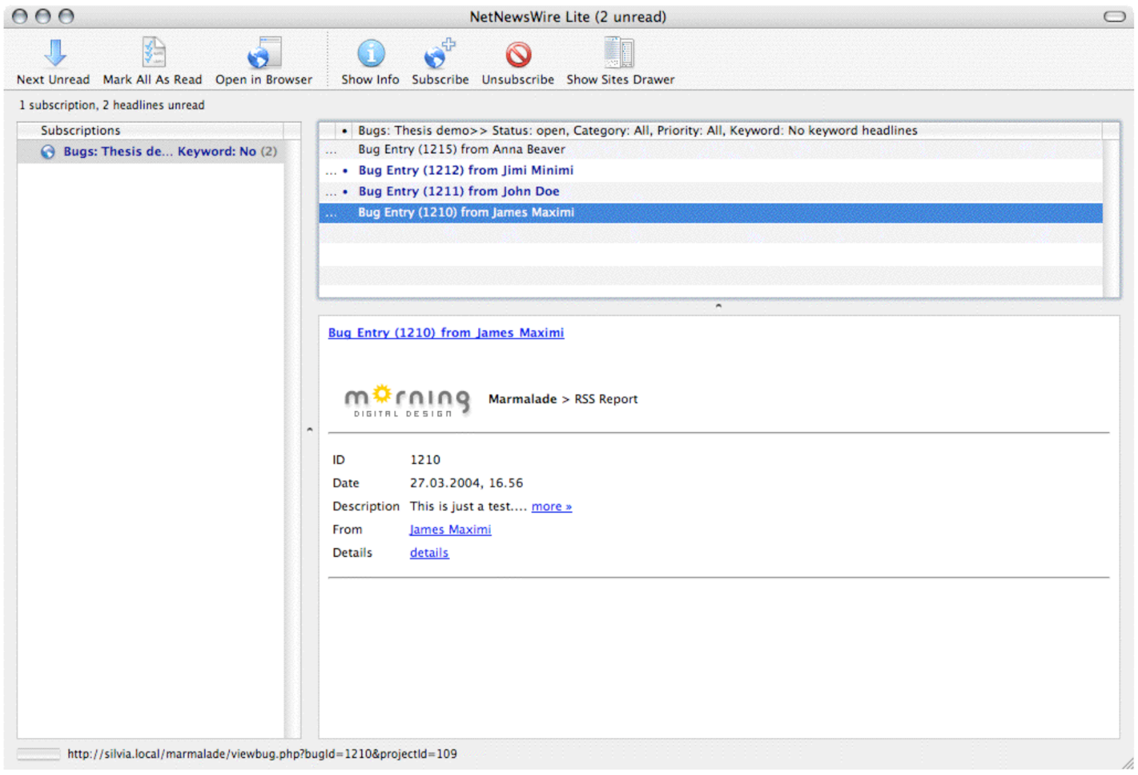
Bug Entry (1210) from James Maximi

Bugs: Thesis demo>> Status: open, Category: All, Priority: All, Keyword: No keyword, 11.4.2004 13:30:43

Marmalade > RSS Report

ID	1210
Date	27.03.2004, 16.56
Description	This is just a test.... more »
From	James Maximi
Details	details

Image 22 – Recommended Windows RSS reader (Wildgrape NewsDesk)



Bugs: Thesis demo>> Status: open, Category: All, Priority: All, Keyword: No keyword headlines

Bug Entry (1215) from Anna Beaver
Bug Entry (1212) from Jimi Minimi
Bug Entry (1211) from John Doe
Bug Entry (1210) from James Maximi

Bug Entry (1210) from James Maximi

Marmalade > RSS Report

ID	1210
Date	27.03.2004, 16.56
Description	This is just a test.... more »
From	James Maximi
Details	details

http://silvia.local/marmalade/viewbug.php?bugId=1210&projectId=109

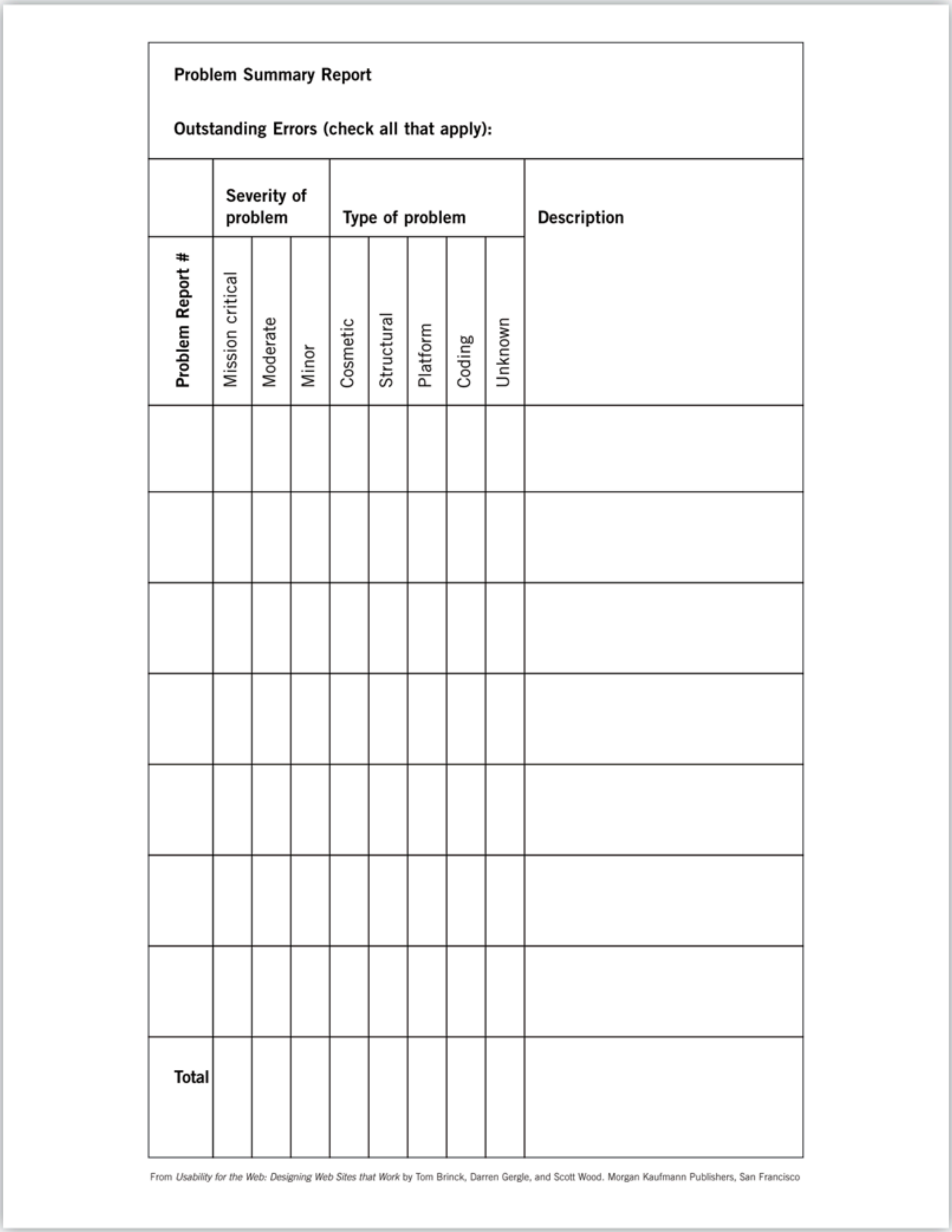
Image 23 – Recommended Macintosh RSS reader (Ranchero Software NetNewsWire Lite)

APPENDIX D: Commercial bug tracking tool license fees

Table 7 - Comparison of commercial bug tracking tool license fees (source: http://www.websina.com/bugzero/bug tracking-pricing.html)

Product	**Company**	**Starting Price**	Expansion Price
TeamTrack 5.6	TeamShare	$4995 (5-user)	unknown
TrackGear 3.0	LogiGear	$1755 (5-user)	$1609 per 5-user
TestTrack Pro 4.7	Seapine	$299 (1-user)	$299 per user
DevTrack 5.0 (Standard)	TecExel	$495 (1-user)	$445 per user
PR-Tracker 5.10	Softwise Company	$199 (1-user)	$199 per user
Problem Tracker 4.1	NetResults	$950 (5-user)	$190 per user
Aardvark 1.15	Red Gate	$1950 (25-user)	$9950 (unlimited)
PVCS Tracker 7.1.1	Merant	$749 (1-user)	$749 per user
TrackRecord _	Compuware	$1495 (1-user)	$1495 per user
Clear Quest	Rational Software	Expensive	Expensive
Bugzero 3.x	WEBsina	$399 (5-user)	$1299 (unlimited)

APPENDIX E: Typical bug reporting forms



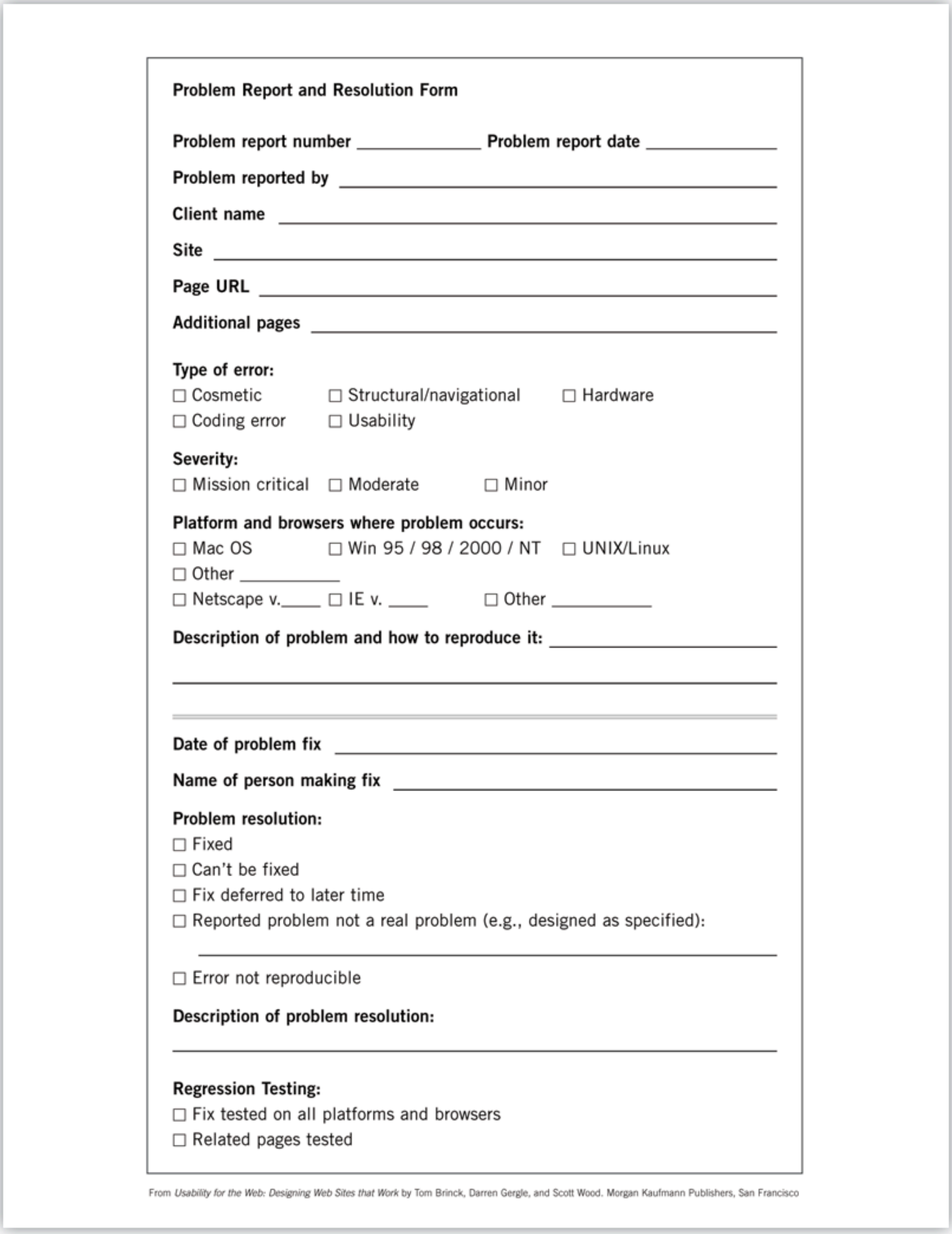
Problem Summary Report

Outstanding Errors (check all that apply):

	Severity of problem			**Type of problem**					**Description**
Problem Report #	Mission critical	Moderate	Minor	Cosmetic	Structural	Platform	Coding	Unknown	
Total									

From *Usability for the Web: Designing Web Sites that Work* by Tom Brinck, Darren Gergle, and Scott Wood. Morgan Kaufmann Publishers, San Francisco

Image 24 – Problem Summary Report



Problem Report and Resolution Form

Problem report number ____________ **Problem report date** ____________

Problem reported by ____________________

Client name ____________________

Site ____________________

Page URL ____________________

Additional pages ____________________

Type of error:

☐ Cosmetic ☐ Structural/navigational ☐ Hardware

☐ Coding error ☐ Usability

Severity:

☐ Mission critical ☐ Moderate ☐ Minor

Platform and browsers where problem occurs:

☐ Mac OS ☐ Win 95 / 98 / 2000 / NT ☐ UNIX/Linux

☐ Other __________

☐ Netscape v.____ ☐ IE v. ____ ☐ Other __________

Description of problem and how to reproduce it: ____________________

____

Date of problem fix ____________________

Name of person making fix ____________________

Problem resolution:

☐ Fixed

☐ Can't be fixed

☐ Fix deferred to later time

☐ Reported problem not a real problem (e.g., designed as specified):

____

☐ Error not reproducible

Description of problem resolution:

____

Regression Testing:

☐ Fix tested on all platforms and browsers

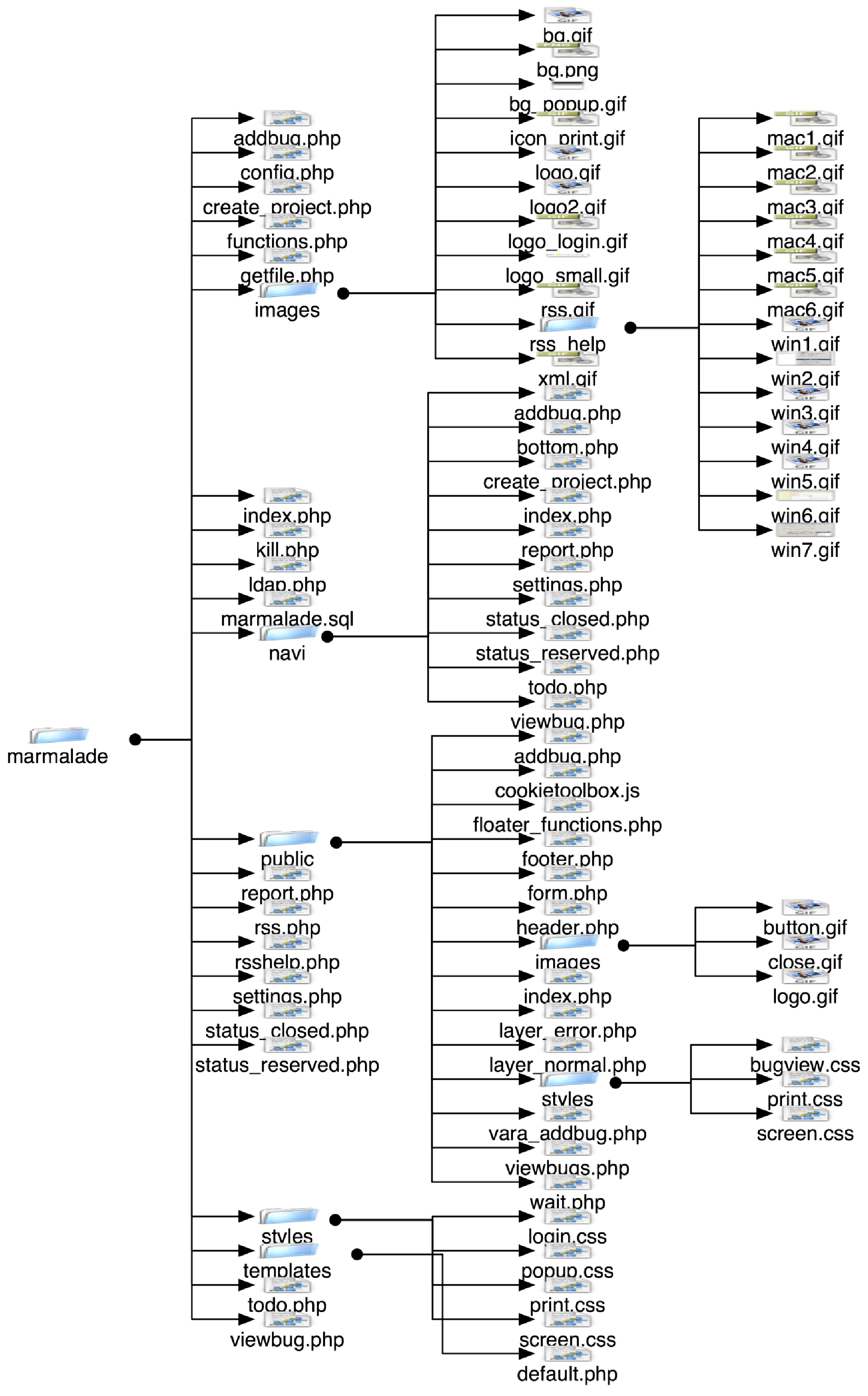
☐ Related pages tested

From *Usability for the Web: Designing Web Sites that Work* by Tom Brinck, Darren Gergle, and Scott Wood. Morgan Kaufmann Publishers, San Francisco

Image 25 – Problem report and Resolution Form

From Usability for the Web by Tom Brinck, Darren Gergle, and Scott Wood. Morgan Kaufmann Publishers, San Francisco.

APPENDIX F: System file structure


marmalade
addbug.php
config.php
create_project.php
functions.php
getfile.php
images
bg.gif
bg.png
bg_popup.gif
icon_print.gif
logo.gif
logo2.gif
logo_login.gif
logo_small.gif
rss.gif
rss_help
mac1.gif
mac2.gif
mac3.gif
mac4.gif
mac5.gif
mac6.gif
win1.gif
win2.gif
win3.gif
win4.gif
win5.gif
win6.gif
win7.gif
xml.gif
index.php
kill.php
ldap.php
marmalade.sql
navi
addbug.php
bottom.php
create_project.php
index.php
report.php
settings.php
status_closed.php
status_reserved.php
todo.php
viewbug.php
public
addbug.php
cookietoolbox.js
floater_functions.php
footer.php
form.php
header.php
images
button.gif
close.gif
logo.gif
index.php
layer_error.php
layer_normal.php
styles
bugview.css
print.css
screen.css
vara_addbug.php
viewbugs.php
wait.php
report.php
rss.php
rsshelp.php
settings.php
status_closed.php
status_reserved.php
styles
login.css
popup.css
print.css
screen.css
templates
default.php
todo.php
viewbug.php