Verso Project

Application Report

Tero Hänninen
Juho Nieminen
Marko Peltola
Heikki Salo

Version 1.0
Public
14.12.2010

University of Jyväskylä

Department of Mathematical Information Technology

Jyväskylä

Acceptor	**Date**	**Signature**	**Clarification**
Project manager	__.__.2010		
Customer	__.__.2010		
Instructor	__.__.2010		

Document Info

Authors:

• Tero Hänninen (TH)	tejohann@jyu.fi	0400-240468
• Juho Nieminen (JN)	juho.nieminen@jyu.fi	050-3831825
• Marko Peltola (MP)	marko.peltola@jyu.fi	041-4498622
• Heikki Salo (HS)	heikki.ao.salo@iki.fi	050-3397894

Document name: Verso Project, Application Report
Page count: 48

Abstract: Verso project developed a web application for source code management and releasing. The document goes through the background of the software, presents the user interface and the application structure and describes the programming and testing practices used in the project. The realization of functional requirements, ideas for further development and a guide for future developers are also presented.
Keywords: Application structure, future development, Git, Gitorious, meeting the requirements, programming practices, Ruby on Rails, software development, source code management, source code releasing, testing, WWW interface.

Project Contact Information

Project group:

Tero Hänninen	tejohann@jyu.fi	0400-240468
Juho Nieminen	juho.nieminen@jyu.fi	050-3831825
Marko Peltola	marko.peltola@jyu.fi	041-4498622
Heikki Salo	heikki.ao.salo@iki.fi	050-3397894

Customers:

Ville Tirronen	ville.e.t.tirronen@jyu.fi	014-2604987
Tero Tuovinen	tero.tuovinen@jyu.fi	050-4413685
Paavo Nieminen	nieminen@jyu.fi	040-5768507
Tapani Tarvainen	tt@it.jyu.fi	014-2602752

Instructors:

Jukka-Pekka Santanen	santanen@mit.jyu.fi	014-2602756
Antti-Juhani Kaijanaho	antti-juhani.kaijanaho@jyu.fi	014-2602766

Contact information:

Email lists	verso@korppi.jyu.fi and yousource-users.group@korppi.jyu.fi.
Email archives	https://korppi.jyu.fi/kotka/servlet/list-archive/verso/ and https://korppi.jyu.fi/kotka/servlet/list-archive/yousource-users.group/.
Workroom	AgC 222.2, tel. 014-2604963.

Version History

Version	Date	Modications	Modifier
0.0.1	20.4.2010	The report template was created.	JN
0.0.2	21.4.2010	The initial table of contents was created and the introduction was written.	JN
0.0.3	22.4.2010	The background chapter was started.	JN
0.0.4	28.4.2010	The background chapter was finished and contents to the user interface chapter were added.	JN
0.1.0	29.4.2010	The user interface chapter was continued.	JN
0.1.1	3.5.2010	Typos were corrected and the structure chapter and the programming practices chapter were started.	JN
0.1.2	4.5.2010	The testing chapter was started.	JN
0.1.3	5.5.2010	The appearance of the document was fixed and the content was modified according to the feedback.	JN
0.1.4	6.5.2010	The realization of objectives was started.	JN
0.2.0	18.5.2010	Figures were added to the user interface chapter. The background chapter was modified. The sitemap and metafiles sections were added.	JN
0.2.1	20.5.2010	Figure labels and figures were modified. Terms were added and clarified. Some text was added here and there.	JN
0.2.2	24.5.2010	The subsections related to the header and sidebar were merged into the page structure chapter. The figure on the inner and external interfaces was added and content for the chapter was written. The sections describing system testing and unsatisfactory solutions in implementations were written.	JN
0.3.0	26.5.2010	The user interface chapter was modified and expanded. The programming practices chapter was written and the realization of objectives chapter was continued.	JN

Version	Date	Modications	Modifier
0.3.1	26.5.2010	The references were modified. The testing chapter was modified and expanded with the testing results section. The realization of objectives chapter was continued.	JN
0.3.2	27.5.2010	The realization of objectives chapter and the testing results section were continued.	JN
0.3.3	28.5.2010	The class structure section was written.	JN
0.3.4	31.5.2010	The signatures table was widened. The document info was added to the document. The chapter on guide for future developers was written. The section on logging in was extended. The figures of YouSource before and after Verso project were added.	JN
0.3.5	1.6.2010	The chapters on introduction, terminology, background and application structure were modified.	JN
0.4.0	2.6.2010	The chapters on application structure, user interface, testing practices, programming practices and realization of objectives were modified. The summary was started.	JN
0.4.1	8.6.2010	The section on inner and external interfaces was continued.	JN
0.4.2	16.6.2010	Minor changes here and there were made according to the feedback. The summary chapter and the modifications during the implementation section were continued.	JN
0.5.0	17.6.2010	Minor changes were done here and there. The figures related to the edit repository page, activities page and manage collaborators page were added. The references were updated.	JN
0.5.1	1.8.2010	Minor changes here and there were done according to the feedback. The term crontab was added. Document keywords were added.	JN
0.6.0	2.8.2010	Minor changes here and there were done according to the feedback. The term hook was added. The section testing results and the chapters on terms and realization of objectives were continued.	JN

Version	Date	Modications	Modifier
0.6.1	8.9.2010	The appearance of the document was improved according to the feedback. Minor changes were made to the content here and there.	JN
0.6.2	9.9.2010	Minor changes were made here and there to the content.	JN
0.7.0	10.9.2010	Typos and wording were corrected on several chapters.	JN
1.0.0	14.12.2010	A keyword 'meeting the requirements' was added. Alternating space between paragraphs was fixed. The description of the term private project was rephrased. Typos were corrected and small changes were made to various parts of the document.	JN

Contents

1 Introduction **1**

2 Terminology **2**
2.1 General Terms 2
2.2 Verso Specific Terms 3

3 Background **5**
3.1 Publishing Channel for Source Codes 5
3.2 Gitorious as a Starting Point 6

4 User Interface **7**
4.1 Sitemap 7
4.2 Page Structure 8
4.3 Logging in 9
4.4 Creating a Repository 11
4.5 Creating a Mirror Repository 13
4.6 Repository Browser 15
4.7 Updating Repository with a Zip Package 16
4.8 Adding Single Files to a Repository 17
4.9 Private Repository 18
4.10 Private Project 19
4.11 Editing of Repository Metafiles 19
4.12 Usability Modifications to the WWW Interface 20

5 Application Structure **22**
5.1 Components 22
5.2 Inner and External Interfaces 23
5.3 Class Structure 24
5.4 Metafiles 26

6 Programming Practices **27**
6.1 Formatting, Naming and Commenting Practices 27
6.2 Source Code Example 28
6.3 Grouping Practices 31
6.4 Development Platform 31

7 Testing Practices and Results **32**
7.1 Integration Testing Practices 32
7.2 Usability Testing Practices 33
7.3 System Testing Practices 34
7.4 Testing Results 34

8 Realization of Objectives **37**
8.1 Realization of Requirements 37
8.2 Unsatisfactory Solutions in Implementation 39
8.3 Challenges in Implementation 40
8.4 Modifications during the Implementation 40
8.5 Further Development Ideas 41

9 Guide for Future Developers **43**
9.1 Essential Bugs 43
9.2 Improvements to Existing Features 44
9.3 The Most Useful New Features 45
9.4 Development Practices 45

10 Summary **46**

11 References **47**

1 Introduction

In a work community, sharing information increases productivity [12]. For this reason Department of Mathematical Information Technology (MIT) at University of Jyväskylä decided to start developing practices for researchers to share their source code with each other. The idea was first tried out by Ville Tirronen who tested a prototype software that did not provide all the functionalities needed by the users. A need for such a software was still recognized but no free or inexpensive solution was found. The idea was then proposed for a student project that was starting in the MIT department.

The project, soon to be known as Verso, developed a web application called YouSource that enables users to share and maintain version history of their source code. The software is based on Gitorious, which is a an open source application for hosting Git repositories. It was chosen to be the starting point of the development because it covered the largest amount of requirements compared to the other reviewed software. Verso project defined, planned and implemented the most essential missing functionalities to Gitorious. The web application was developed with Ruby on Rails framework to be run on Linux servers and PCs and viewed by any modern web browser.

Verso project wrote several documents describing the developed software and the the project itself. The realization of the goals and the practices are described in the project report [10]. The usability testing sessions are described in the memos [8] and [9]. The system testing practices and the test cases are described in [3] and the results in [4] and [5]. Possible software platforms were compared in the beginning of the project and the results were reported in [6].

The document describes the implementation of the features described in the requirements specification [2]. In Chapter 2 the essential terminology used in the document is explained. Chapter 3 describes the background of the project. Chapter 4 presents the user interface of the application concentrating on the parts that were added and modified in the project. Chapter 5 explains the inner structure of the application. Chapter 6 specifies the programming practices used during the project and Chapter 7 describes how testing was carried out. In Chapter 8 it is considered how the objectives set to the application were fulfilled. Finally, in Chapter 9, suggestions on how to start the further development are presented.

2 Terminology

The chapter explains the essential terms that appear in the document.

2.1 General Terms

The following terms are related to the software that was developed in Verso project but they are non-specific and could apply to many other applications as well.

Branch in Git is a pointer to a commit. The current branch determines where the user's new commits will go. A branch is considered as a course of development.

Commit contains file modification data and a log message describing the changes that are produced by the user .

Crontab is a configuration file for using a program called Cron. Cron is a time-based job scheduler that can be set to execute shell commands periodically.

Hook is a piece of software code that handles intercepting function calls or messages between software components.

Metadata is information about data. It describes the definition, structure and management of data files.

MVC architecture i.e. Model-View-Controller, is an architectural pattern used in software engineering. The pattern isolates the application logic from the input and the presentation, permitting independent development, testing and maintenance of each.

Push uploads the new commits in the local repository to a remote repository. It can be thought as releasing a version of a code if one is pushing into a public repository.

Ruby on Rails is an open source web application framework for Ruby programming language.

2.2 Verso Specific Terms

The terms below are specific to Verso project and describe some of the key elements in the developed software.

Owner is able to manage the data of one or many projects which in turn have one or many repositories. The owner of a project is also the owner of the repositories in the project. An owner is allowed to commit to his repositories and modify the data in his projects and repositories. An owner is a user or a team.

Private project can only be seen by the owner of the project, the members of the team that owns the project, and the users that have view, review, commit or administrate rights to one of the repositories in the project.

Private repository can only be seen by the users that have view, review, commit or administrate rights to it.

Project is a user created entity in YouSource which has one or many repositories. It can have many attributes, for example a description, a home page URL and keywords. A project is owned by a user or a team.

Public project is visible to all visitors of the web page.

Public repository is visible to all visitors of the web page.

Repository refers to a Git repository which belongs to a project. It stores data and the version history of the data. It can have several attributes, for example a description and a license. A repository is owned by a user or a team.

Team is a group of users. At least one member in the team has admin rights and the rest are normal team members.

Viewer can see a repository even if it is marked as private. The wiewer right is the lowest of the possible rights to a repository. The others are in order review, commit and administrate.

YouSource is the name of the application developed by Verso group.

3 Background

Verso project was started off from a need for a better way to share source code inside a work community and an interest in a prototype software for that purpose. The chapter goes briefly through the background and the needs of the developed software as well as what led to starting of Verso project and the development of YouSource. Also the initial needs are shortly presented.

3.1 Publishing Channel for Source Codes

Currently, at Department of Mathematical Information Technology in University of Jyväskylä, there is no commonly used practices for **storing and releasing source code**. This causes a number of problems. For instance, when a worker leaves the department all the source code he has developed is often lost too. Poor practices on communication prevent workers from knowing who is doing what, which may lead to producing overlapping work. Furthermore, the current disorganized practices present licensing problems in which one is unaware who owns a piece of source code and how it can be used.

The challenge and the aim of the department is to get as many employees as possible to use proper **version control**. Therefore, a software was needed that can be used in many different ways to support different users. A WWW interface was needed especially for the employees and students unfamiliar with version control. A command line interface was needed for the more experienced. Even features that will adapt to the current unique working methods of the employees should be implemented, such as reading and mirroring a zip archive at a supplied URL.

Prior to Verso project there was a **prototype** that was tested by Ville Tirronen but it didn't meet the needs of the users. The task of developing another prototype was given to Verso project because the department has a clear need of improving the way the employees share their work. Tirronen was one of the customer's representatives and guided the project with his experiences from the previous prototype.

3.2 Gitorious as a Starting Point

When the key functional requirements for the needed software were put together, it soon became apparent that software closely fitting for the requirements already existed. One noted application was GitHub which basically covered all the key requirements. However, GitHub is not open source and buying it would be too expensive. More alternatives were reviewed and compared in [6] and finally Verso group ended up with Gitorious described in [11].

Gitorious is an open source application for hosting projects that use Git. It stores users' repositories and provides useful tools to manage them. Gitorious encourages users to collaborate with each other which gives it a feel of a social networking website. Other reviewed software that came close to the initial requirements were FusionForge, InDefero, Savane, Project Kenai, Fedora Hosted and KnowledgeForge. The disadvantages with these were lack of recent development, difficult repository creation process, no activity view for repositories and/or limited search functions.

Gitorious had most of the required key features already implemented. It supports a widely used version control application called Git, it has a WWW interface that covers most of the key functionalities and all the information about the projects and repositories is easily obtainable. However, some of the requested features were missing from Gitorious. It didn't support private projects or repositories, it didn't save project information (metadata) to the repository itself and it didn't support any update methods other than Git. All these features were added to YouSource by Verso group. In addition, the logging in was changed to be similar to the other services of the university and usability was increased with user interface modifications.

4 User Interface

The user interface of YouSource was developed using HTML elements and a cascading style sheet (CSS) file. The general look of Gitorious was left untouched but almost all the new features needed additions and modifications to the views. Not only the web pages related to the new features were changed but also minor changes, such as adding links and changing labels, were made here and there to improve usability. The chapter describes the user interface changes that were done to Gitorious by Verso group.

4.1 Sitemap

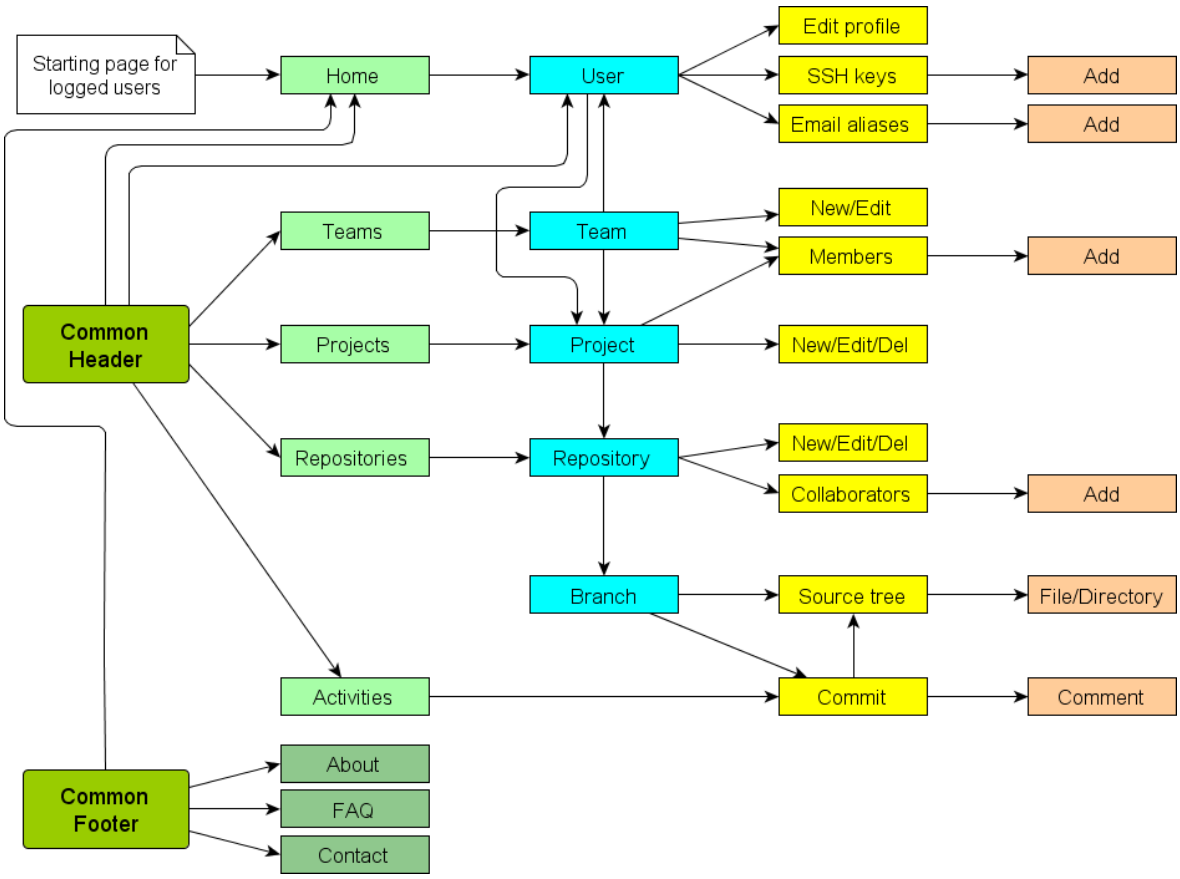

Figure 4.1: Simplified sitemap of YouSource.

Since YouSource is based on a fully functional web application called Gitorious it has a lot of pages. Figure 4.1 shows nearly all the pages and describes their relations. Not all links are visible to simplify the picture.

The common header and common footer components are present in every page and provide links to the low level pages of the application. In Figure 4.1, the horizontal axis displays the relative path of the pages and the vertical axis displays the logical dependencies between the pages. For instance, a project has many repositories and a branch has many commits.

In Figure 4.1, the starting page for a logged user is shown to be *Home* page. However, for a non logged user the starting page is *Activities* page (see Figure 9.1) where a lot of recent site activity is shown. The non logged user can explore the site and visit public projects and repositories and their subpages but he can not create any new projects or repositories.

4.2 Page Structure

A general page structure in YouSource consists of five elements: the header, breadcrumb, main content, sidebar and the footer (see Figure 4.2). The header, footer and the breadcrumb sections offer navigation and user management.

The **header** is located at the top of each page and contains the main menu and the user menu. The main menu contains links to all the main pages of the application and the user menu contains links to user information and user management. For users not logged in the user menu contains only a link to the login page.

The **breadcrumb** navigation is located right under the header but it is not visible on all pages. It contains the logical path to the currently visible page, for example *project / repository / branch / source tree*. The breadcrumb is not visible on the four main pages *Activities, Projects, Repositories* and *Teams* or on the project creation and the team creation pages.

The **main content** presents the essential information on the page like the site and project activities. All the forms that are used in creating and managing projects, repositories and teams are also shown in the main content area as well as all the warning and notification messages.

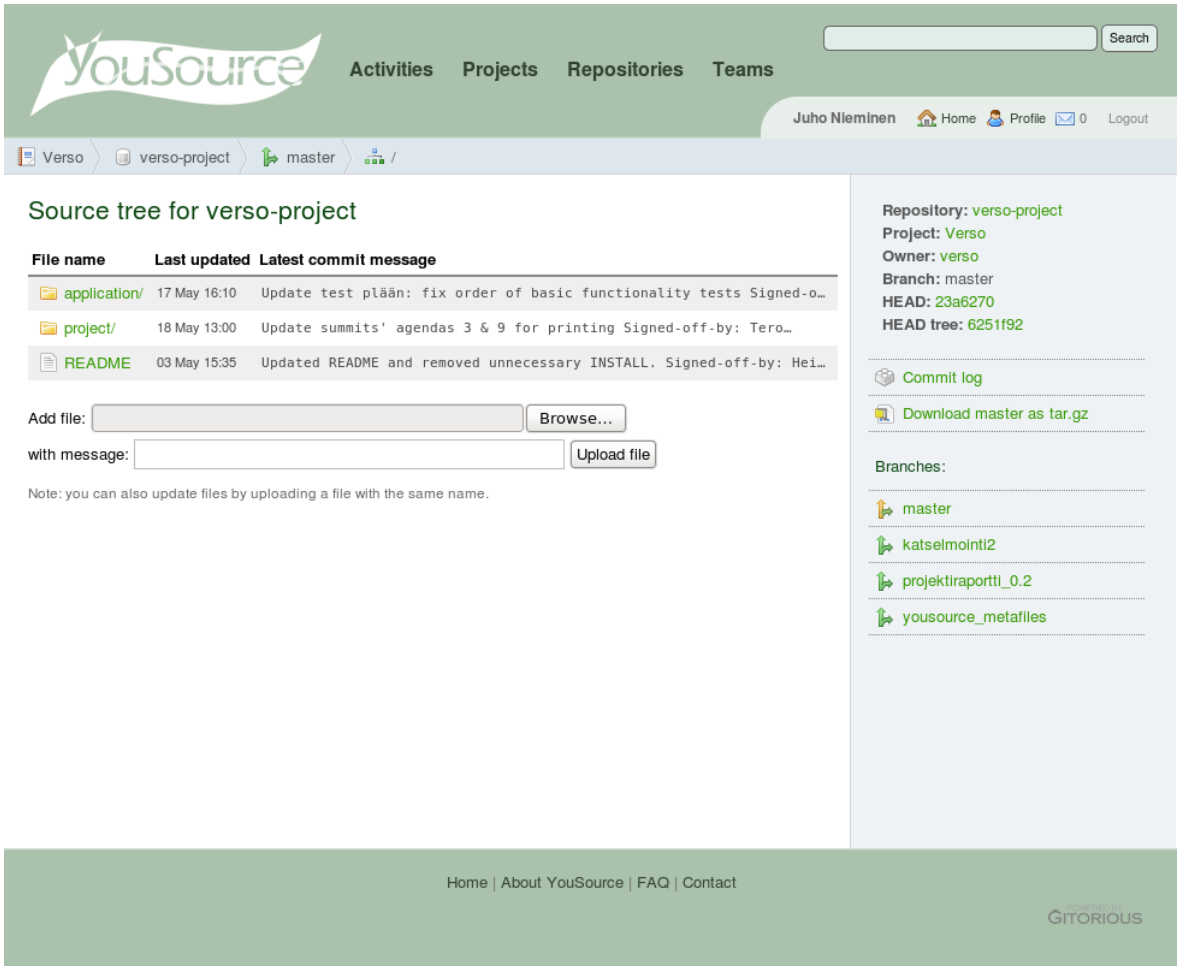

Figure 4.2: The header, breadcrumb, sidebar, main content and the footer.

The **sidebar** is visible on most of the pages in the application. It contains additional information for the main content section and links to management pages.

The **footer** is located at the bottom of each page. It contains a menu with links to the informative pages *About*, *FAQ*, and *Contact*. A full view of a page in YouSource that displays all the elements described in the section is shown in Figure 4.2.

4.3 Logging in

Gitorious uses an email address and password pair for logging users into the system. This was considered inconsistent with most of the other web applications in University of Jyväskylä as they use a **username and password pair for logging in**. Therefore, YouSource was changed to use the username too instead of an email.

One of the main information systems in the university is Korppi, which provides various kinds of services for students, employees and guests. Korppi also provides an **LDAP authentication interface**, and this was used to log users in to YouSource. The LDAP authentication made it possible to not store user passwords at all to YouSource, since Korppi already has that information. This led to removing all the UI elements that handled passwords except for the login page which is shown in Figure 4.3.

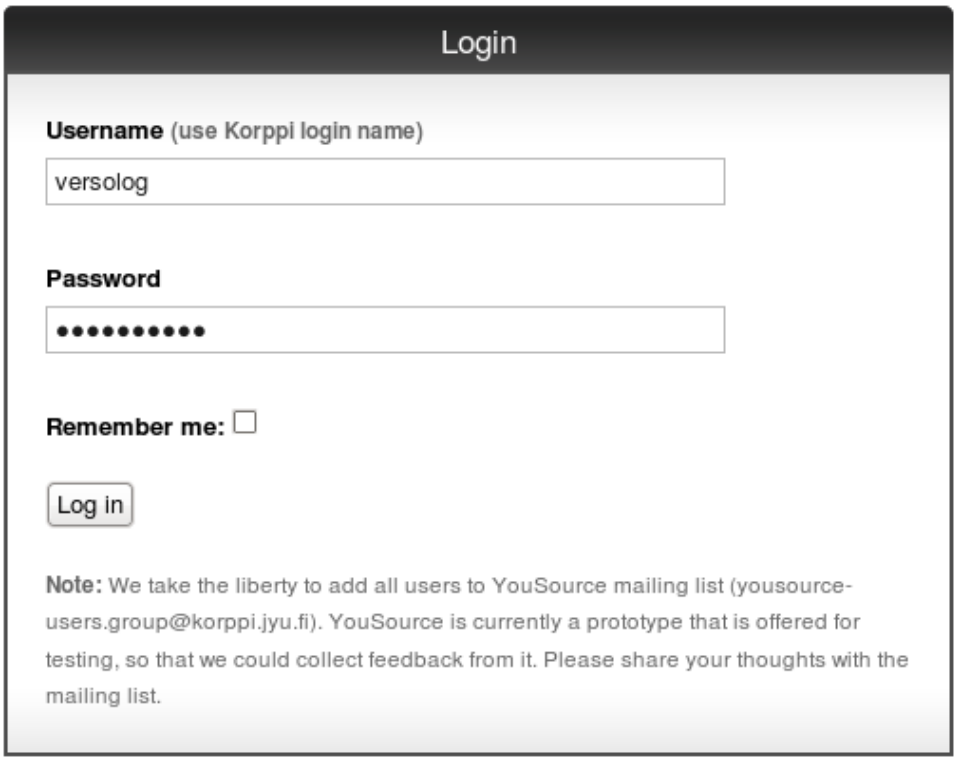

Figure 4.3: The login view.

Gitorious sends an activation email after a new user has registered. This feature was removed from YouSource since it is not needed because users can't register new usernames to the site by themselves. Instead, after the first login the site itself creates a new user to its database with the provided username, if the username and password matched the Korppi database.

When a user creates a new account in Gitorious, he must accept the current **terms of service**. In YouSource the terms of service were removed because the terms policy wasn't clear to the customer at the time. However, since the terms of service might be used in the future, the user entries in the database have the terms of service field and it is marked as accepted for the users for now. Therefore, no users who

logged in to YouSource and had the terms of service accepted for them should be held accountable for any future terms of service the application might have.

4.4 Creating a Repository

Figure 4.4 displays the **form that is used to create a new repository**. In *Basic information* section the user specifies the project in which the repository will be created and the name and the description of the repository. The description is an optional attribute. *Options* section lets the user further specify some optional features for the repository. These are initializing the repository with a local zip file, setting a mirror repository (more of this in Chapter 4.5), marking the repository as private and enabling merge requests from other users.

The way a user can create a repository has changed a lot from Gitorious to YouSource. Gitorious offered only one way to **create a repository** while in YouSource a user has **several ways** to do it. A user can initialize the repository with a zip file uploaded from his computer, or he can set up a mirror repository by providing a URL to a zip file or an SVN repository. If the user prefers not to use any of these options, a normal empty repository will be created.

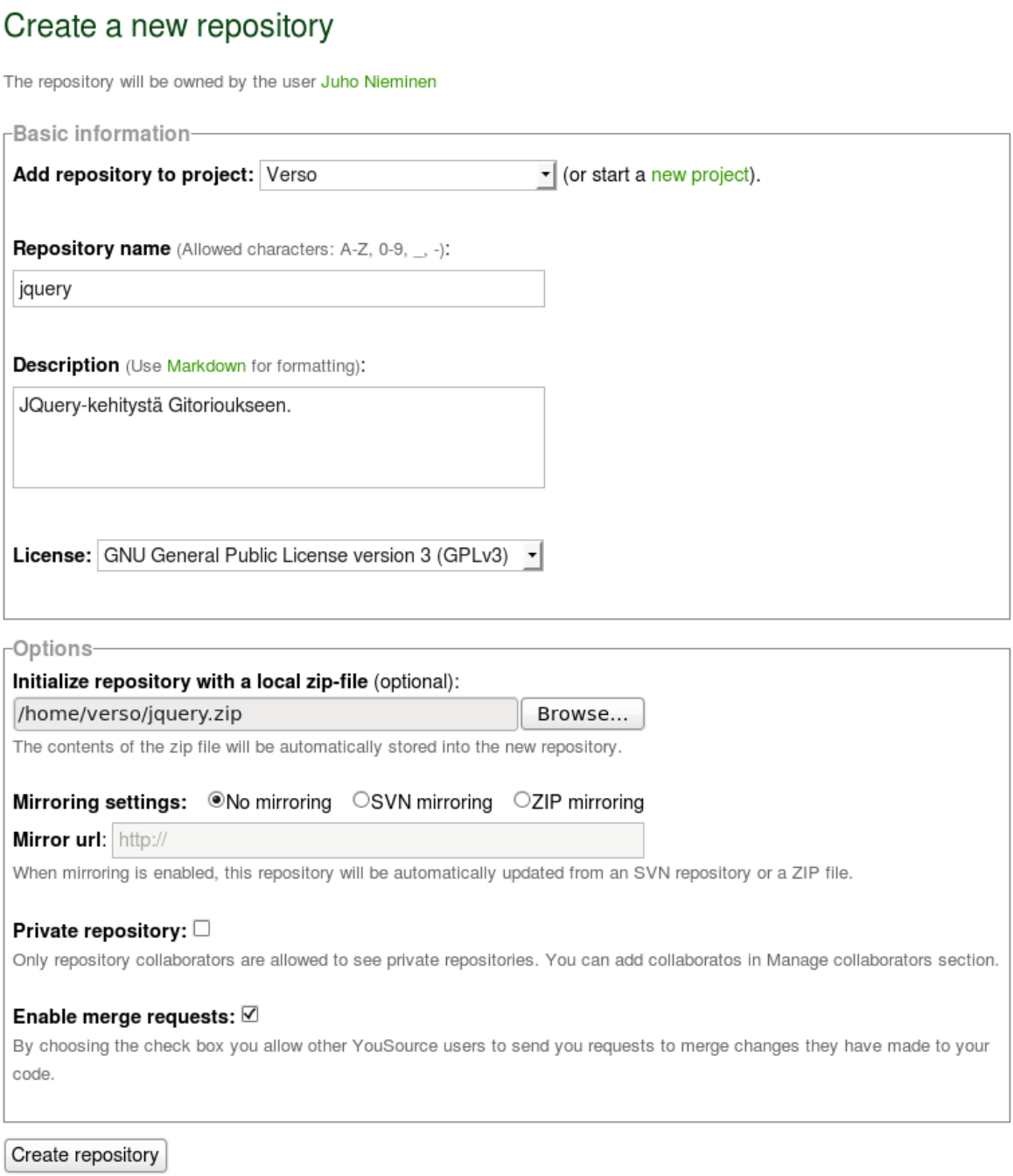

Figure 4.4: The form for creating a new repository.

4.5 Creating a Mirror Repository

While creating a repository it is possible **to provide a URL** that specifies a zip file or an SVN repository in the *Options* section of the repository creation form (see Figure 4.4). This URL is called a mirror URL and it will be stored as an attribute to the repository. If a URL for a zip file is provided the repository will be created normally with Git and the contents of the file will be added into it. On the other hand, if the user selects the SVN mirroring, the repository will be created by cloning the SVN repository.

If a mirror URL has been specified for a repository, the contents of the zip package or the SVN repository **will be downloaded daily** to the repository in YouSource. The script is scheduled in crontab to run every day at seven o'clock in the morning. It will go through all the mirror repositories in YouSource and checks if there are any changes in the source file or in the repository at the mirror URL. The mirror URL can be changed at any time on the repository edit page (see Figure 4.6) which can be accessed through the sidebar of the repository page (see Figure 4.5).

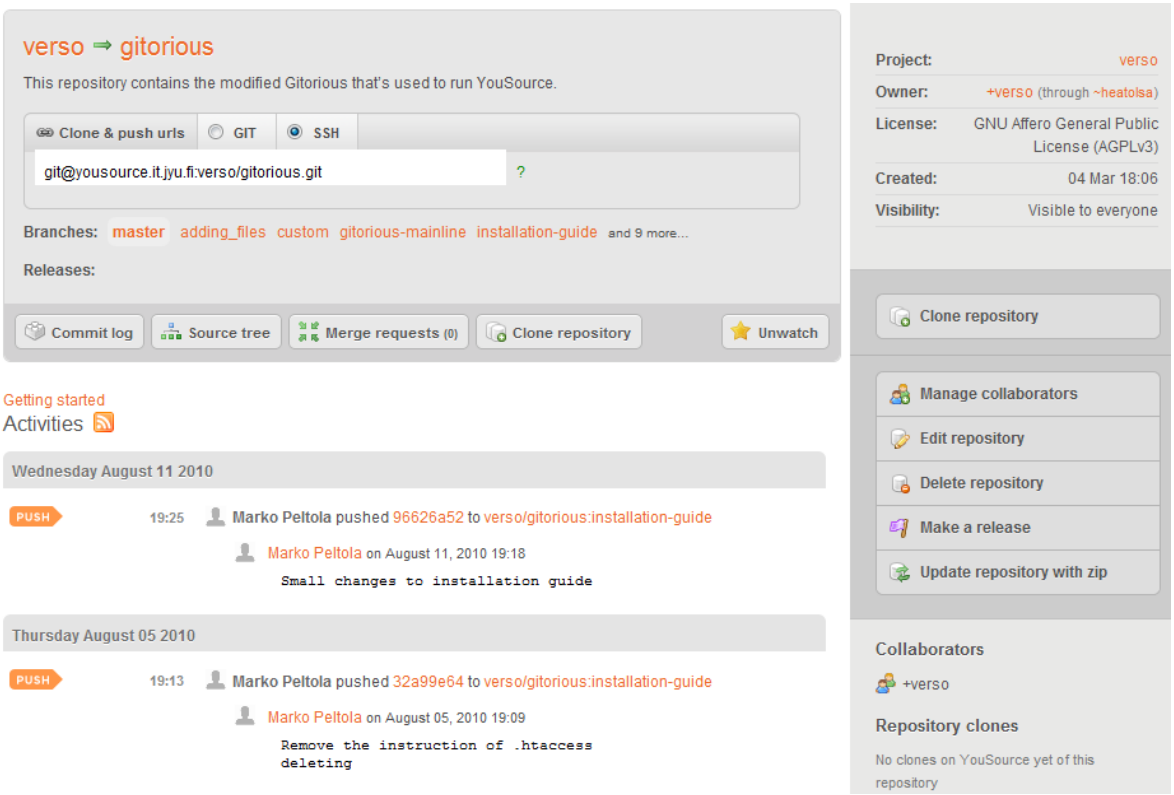

Figure 4.5: The *Repository* page. The image was taken after some of the page styles were altered.

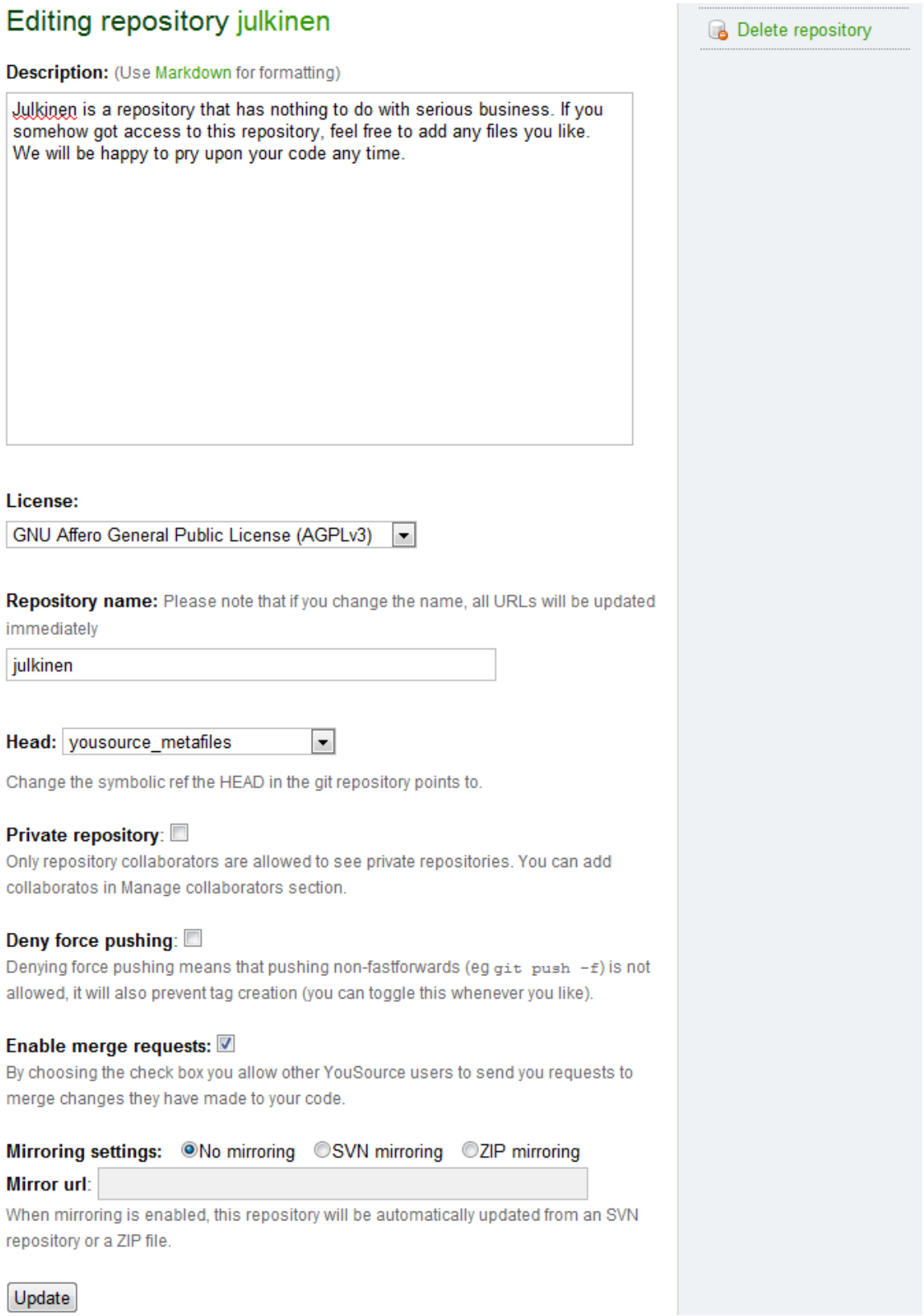

Figure 4.6: The *Edit repository* page.

4.6 Repository Browser

Gitorious doesn't offer a listing page for repositories as it does for projects. In YouSource this was corrected and a page displaying all the public repositories on the site was created. In YouSource the page is one of the four main pages which are *Activities*, *Projects*, *Repositories* and *Teams*.

The *Repositories* page in Figure 4.7 consists of a list of repositories and a side bar. The **repository list** is in the order the repositories are updated (latest first). The list is divided into pages so that only twenty repositories are shown at once, and the rest can be viewed through page number links at the bottom of the list. If a user is logged in, the repositories in which the user has commit rights are highlighted with a slightly darker background color as is seen in the two first repositories in Figure 4.7.

In the **sidebar** there is a list of the most active repositories from the past two weeks and a list of the user's own active repositories (if logged in). The sidebar also provides a link to create a new repository.

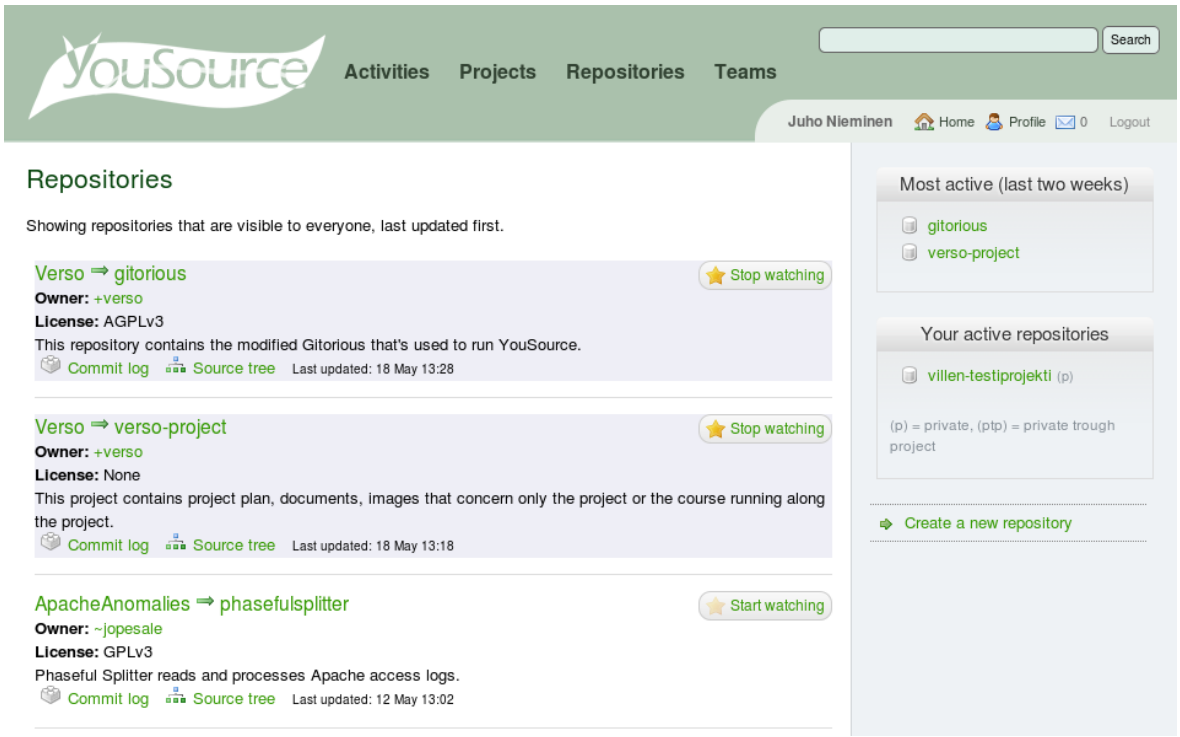

Figure 4.7: The repository browser (*Repositories* page).

4.7 Updating Repository with a Zip Package

On the repository page (see Figure 4.5) the sidebar contains links to operational pages related to the repository. The link *Update repository with zip* leads to a form on the page presented in Figure 4.8. The form accepts a URL of a zip file or a path to a local zip file for the update process. In the form the user also specifies the branch into which he wants the zip file to be pushed (updated). The default branch is the master branch if that exists. Otherwise, the default is a new branch but the name for the new branch is suggested as *master*.

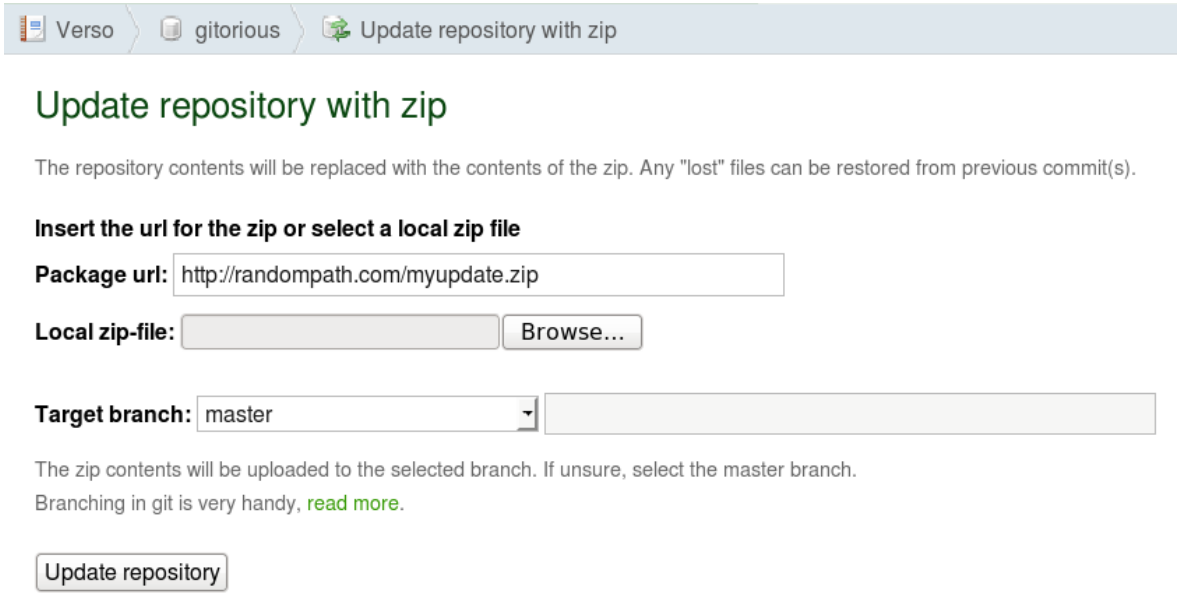

Figure 4.8: The view of updating a repository with a zip file.

4.8 Adding Single Files to a Repository

Gitorious offers a way to browse the contents of a repository with a source tree view (see Figure 4.9). At the bottom of the page there was added a small form which accepts a local file. After sending the form the file will be added to the current directory shown in the breadcrumb. The file will be added into the branch where the repository head is currently in. The form also allows updating a file by specifying an already existing file name in the repository.

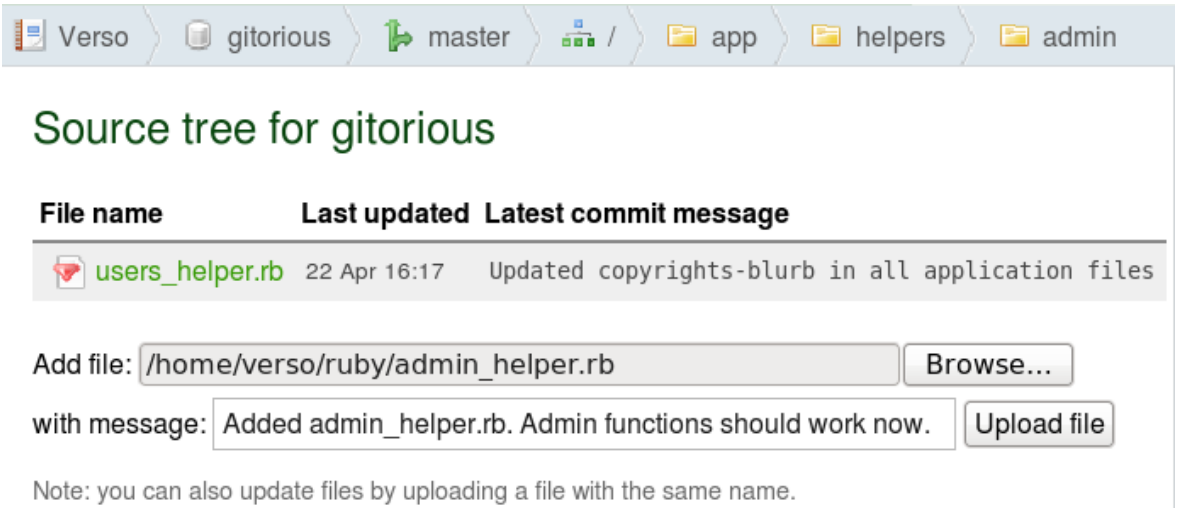

Figure 4.9: The form for adding single files to the current branch.

4.9 Private Repository

A repository can be marked as private (as opposed to public) when the repository is being created (see Figure 4.4 in Section 4.4) or later on the repository edit page (see Figure 4.6). A private repository means that it will only be shown to the users who have view rights to it.

A viewer is a new type of user role that was added into YouSource. The owner of a repository can modify who can view his repository in *Manage Collaborators* page (see Figure 4.10). The view rights can be given to users and teams. Latter will set all the members of the team as viewers. However, a viewer can only access the repository page (see Figure 4.5), not the repository data. To access the data, the committer or administrator right to the repository is required.

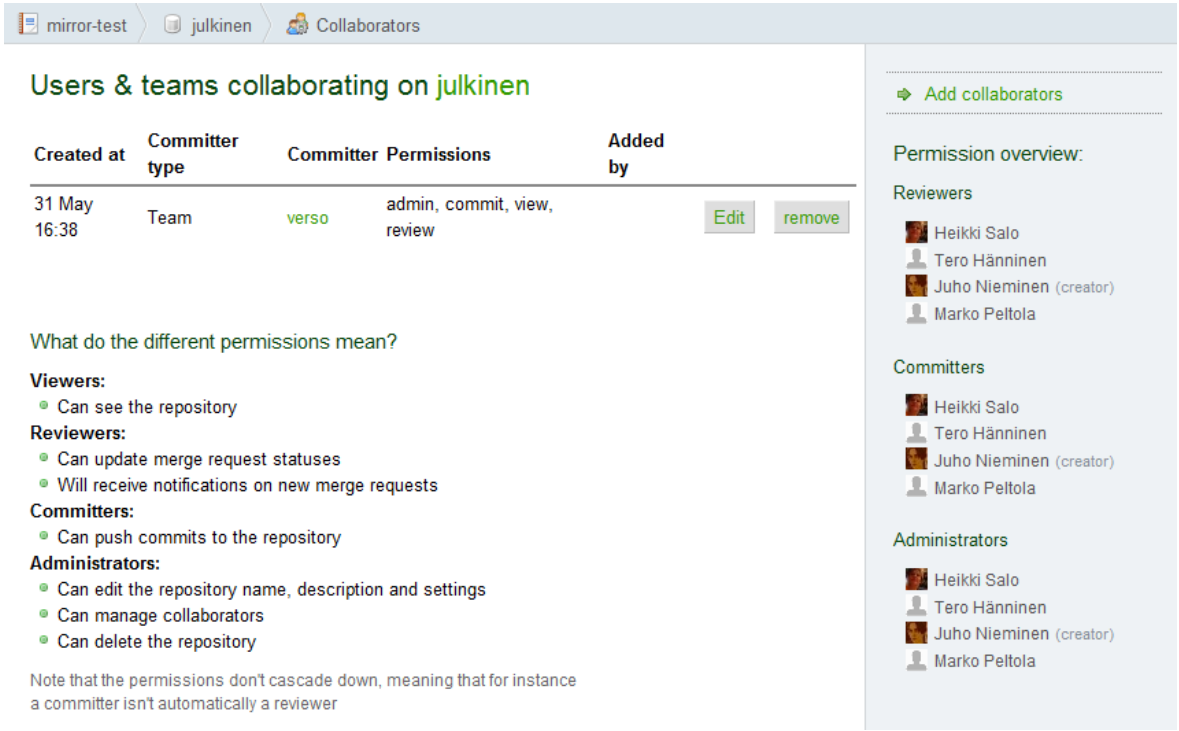

Figure 4.10: The *Manage Collaborators* page.

4.10 Private Project

Like repositories, projects can be marked as private as well. This can be done when the project is created (similar to Figure 4.4 in Section 4.4) or later on the project edit page (similar to the page in Figure 4.6 in Section 4.5). However, projects have a **three step privacy** while repositories have two step privacy. A project can be set to be visible to everyone, visible to users that are logged in or visible only to the members of the project (see Figure 4.11).

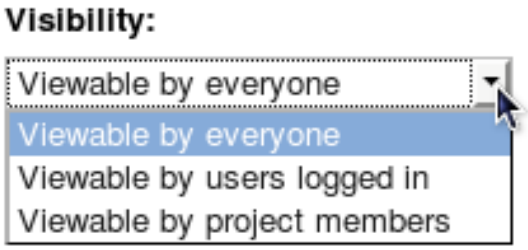

Figure 4.11: The options in creating a private project.

Members can be added and removed on the project members page which can be accessed through the sidebar on the project page. It should be noted, that a project is visible to both to the members and to the users who have gained view rights to one of the repositories in the project. The non-members still can't view the other repositories in the project.

4.11 Editing of Repository Metafiles

Not all user interface changes were related to the WWW interface. One improvement was implemented to the use of the **command line interface** with YouSource. The repository name, description and the option to allow merge requests are also editable through the command line interface. This was achieved by adding a branch called `yousource_metafiles` to every repository created in YouSource. The branch contains a plain text file specifying the description and a YML file containing the other options (see Section 5.4). These files are synchronized with the repository options so that changes in the mentioned files show up in the repository edit page (see Figure 4.6 in Section 4.5) and vice versa.

4.12 Usability Modifications to the WWW Interface

One of the main problems with the prototype software that preceded YouSource was the user interface. It was not clear enough for the users and it didn't encourage employees to use the application. Due to this the customer in Verso project wanted the user interface of the new software to be developed more user friendly than the prototype. Partly, this is taken into account in the requirements for features enabling easier update methods and user logging but usability improvements were needed also in the build-in features of Gitorious.

Usability issues were discovered in various ways. Verso group found problems on their own by using the application during the development. Verso group also consulted a usability expert Meeri Mäntylä and got valuable feedback from a couple of initial users, the instructor Jukka-Pekka Santanen, the usability testing sessions and from the system testing.

The most notable changes where made to the **header and footer** from which the header was totally redesigned. The menu in the header was centered and the user management links were separated from the main menu. Also, Auri Kaihlavirta supplied a logo, a color theme and a bookmark icon for the website. The new header is shown in Figure 4.2 in Section 4.2 and for comparison the Gitorious header is shown in Figure 4.12.

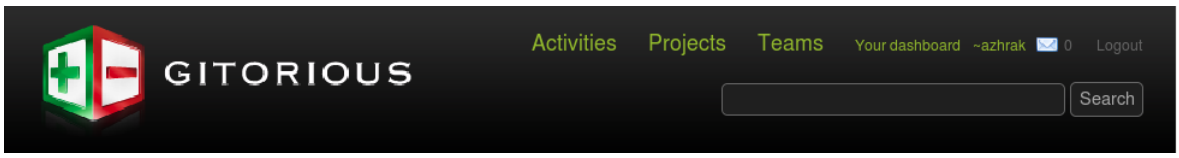

Figure 4.12: Site header of Gitorious when logged in.

YouSource relies on **SSH key authentication**. Users are asked to upload a public SSH key before they can make a project. The SSH key generation process proved to be difficult for inexperienced users during usability testing [9]. For this reason, the SSH key help was remarkably improved.

The usability testing session, described in [8], brought up another issue concerning a help box. In Gitorious, when a user creates a new repository, a **getting started message** is shown until an initial commit is made to the repository. After that there is no way to bring the useful getting started message back. In YouSource this was

corrected so that the getting started button is always shown and the getting started message can be viewed any time by clicking the button.

Meeri Mäntylä reviewed YouSource and gave instructions to the project group how to improve the usability of the application. Based on her advice, Gitorious' term *dashboard* (user's home page) was changed to the term *home*. Form labels were modified so that they have a colon at the end (e.g. *Project name:*). The difference between a project and a repository was increased by adding a small label on top of the project and repository names to indicate which one is in question.

Verso group found a few usability flaws in Gitorious during their own use of the software. The **buttons** to create a new project and a new repository were added to project and repository pages respectively. Buttons to delete a project or a repository were added to their own pages. The **confirmation message** for deleting a project was unified with the repository deletion confirmation.

One of the instructors in Verso project, **Jukka-Pekka Santanen**, tested YouSource in the final stages of development. He suggested several usability improvements to the WWW user interface. The *Create a new repository* page (see Figure 4.4 in Section 4.4) in particular was modified to be more self-explanatory based on Santanen's feedback. Many other pages too were improved according to his suggestions. To mention a few, *Projects* and *Repositories* pages (see Figure 4.7 in Section 4.6) got a clarifying hint stating the order of the items listed on the pages. The directory table in *Source tree* page (Figure 4.2 in Section 4.2) was supplemented with headings. In the **sidebar** of the *Repository* page the link *Repository clones* was made to be clickable only if clones exist.

5 Application Structure

The chapter describes the different components in YouSource and their relations to each other. YouSource is mainly a server application because of its web service nature. The application uses Ruby on Rails libraries, MySQL database, UltraSphinx search engine, Stomp queuing server, Poller daemon to execute the queues, Git daemon for file download service and a few external Ruby libraries (most notably Grit to handle Git functions).

5.1 Components

Git daemon is a simple server for Git repositories. It uses TCP and listens to a single port and waits for a connection asking for a service and delivers that service if it is enabled. Git daemon makes it possible for users of YouSource to push into repositories and clone them with an URL such as `git://versotest.it.jyu.fi/~verso/gitorious.git`.

Grit provides object oriented read and write access to Git repositories via Ruby. YouSource uses Grit in most of its Git operations. Grit was developed to power GitHub [1], a source code management website very similar to Gitorious.

KorppiLDAP is an authentication and directory access interface. It is used in YouSource during the login process for authenticating the users with Korppi credentials.

MySQL is a relational database management system. It grants access to YouSource's database which stores all the data related to the website such as information related to users (except passwords), events, projects and repositories.

Poller daemon is a script that is used to execute commands from Stompserver's queue. Actions that rely on poller are merge request handling, repository creation, archiving and deletion, SSH key handling, Git functions and email notifications. The poller is always running because without it, none of these actions would be executed.

Ruby libraries are used to perform some specific functionalities that Ruby on Rails doesn't provide. In addition to Grit, the most used libraries in Verso project were **Diff:Display** to display Git version comparisons, `git diffs`, elegantly and `Validates_URL_format_of` to check the correctness of a user specified URL.

Ruby on Rails libraries provide the basic functionalities for many classes in YouSource by inheritance. For instance, the controller classes are inherited from `ActionController` class, the model classes are inherited from `ActionRecord` class and the processor classes are inherited from `ApplicationProcessor` class.

Stompserver is a Stomp messaging server with file, database, memory or activerecord based first-in-first-out (FIFO) queues, queue monitoring and basic authentication written in Ruby programming language. YouSource uses the queue for actions described in Poller daemon.

UltraSphinx is a Ruby on Rails configurator and client to the Sphinx full text search engine. YouSource uses UltraSphinx for the search in the header of each page. It provides a text search of many attributes such as project and repository names and descriptions.

5.2 Inner and External Interfaces

Let us examine the interfaces used in YouSource from **inside the application**. The interfaces fall into categories of inner and external as presented in Figure 5.1. Ruby on Rails libraries and Ruby libraries belong to the inner interfaces and everything else is an external component. In the figure, the web application represents all the classes and modules used in YouSource.

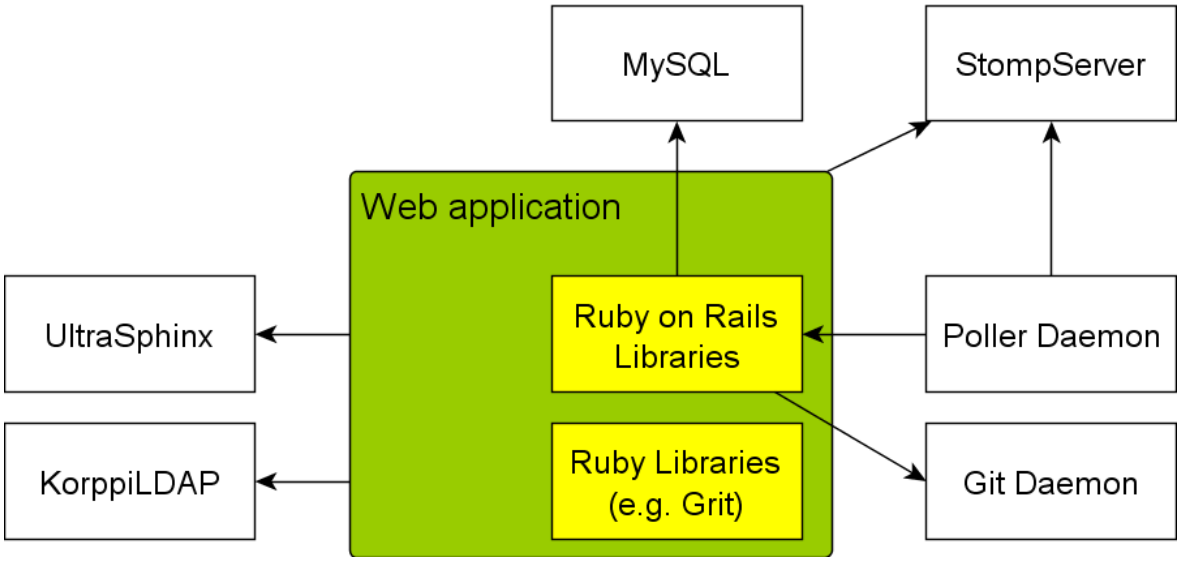

Figure 5.1: The components used in YouSource and their relations to each other.

Examing the interfaces outside from the **user's point of view**, the picture looks a bit different. Interfaces that appear as external for the user are HTTP, SSH and Git. Nothing else is visible to the user which means all the other interfaces are inner interfaces.

The external interfaces used in YouSource are the same as in Gitorious except for KorppiLDAP and Git daemon. **KorppiLDAP authentication** is handled with a Python script. `sessions_controller.rb` receives a username and a password and asks from the class `user.rb` if they are a match. `user.rb` passes the username and password to the Python script `ldapauthenticate.py` which handles the LDAP connection and authentication and returns true or false to indicate whether the authentication was successful or not.

The interface between **Git daemon** and YouSource was modified so that it checks the format and existence of the metafiles. Modifying repository metadata through the Git interface was a new feature developed in Verso project which is done by modifying the metafiles of a repository. When a `git push` is received the hook `data/hooks/pre-receive` performs checks to the incoming data. For the metafiles, it uses the class `data/hooks/pre_receive_guard.rb` to verify if the metadata is valid. In case of an error, the `pre-receive` hook rejects the push and displays an error message.

The **alternative update methods** (see Sections 4.5, 4.7 and 4.8) developed to YouSource required more changes to the inner interface between Git and YouSource. The class `repository.rb` handles the transactions between the two with an external library component Grit if possible, and otherwise directly with command line execution commands of Ruby. Operations handled with the latter method are updating a repository with a zip package, creating a repository with provided zip package as well as creating and updating an SVN clone repositry. Updating the metafiles through the WWW interface and adding a single file to a repository are done with Grit.

5.3 Class Structure

The class structure of YouSource is presented in Figure 5.2. It shows the `Repository` model and the most essential classes related to it. The other models in YouSource like `User`, `Team` and `Project`, have similar structures. In the figure, the classes

marked with a light color, `ActiveRecord.rb` and `ApplicationController.rb`, indicate Ruby on Rails base classes. The classes marked with a darker color, such as `Repository.rb` and `RepositoriesController.rb`, indicate classes that were modified during the project.

The class structure in YouSource is dictated by Ruby on Rails. Rails applications use the **Model-View-Controller** (MVC) architecture, which means that classes are divided into application logic (`model`), input handling (`controller`) and data display classes (`view`). Besides models, controllers and views, there are `processor` classes that handle queued tasks triggered by controllers and `helper` classes that help views to process display data. Other types of classes are used from external libraries.

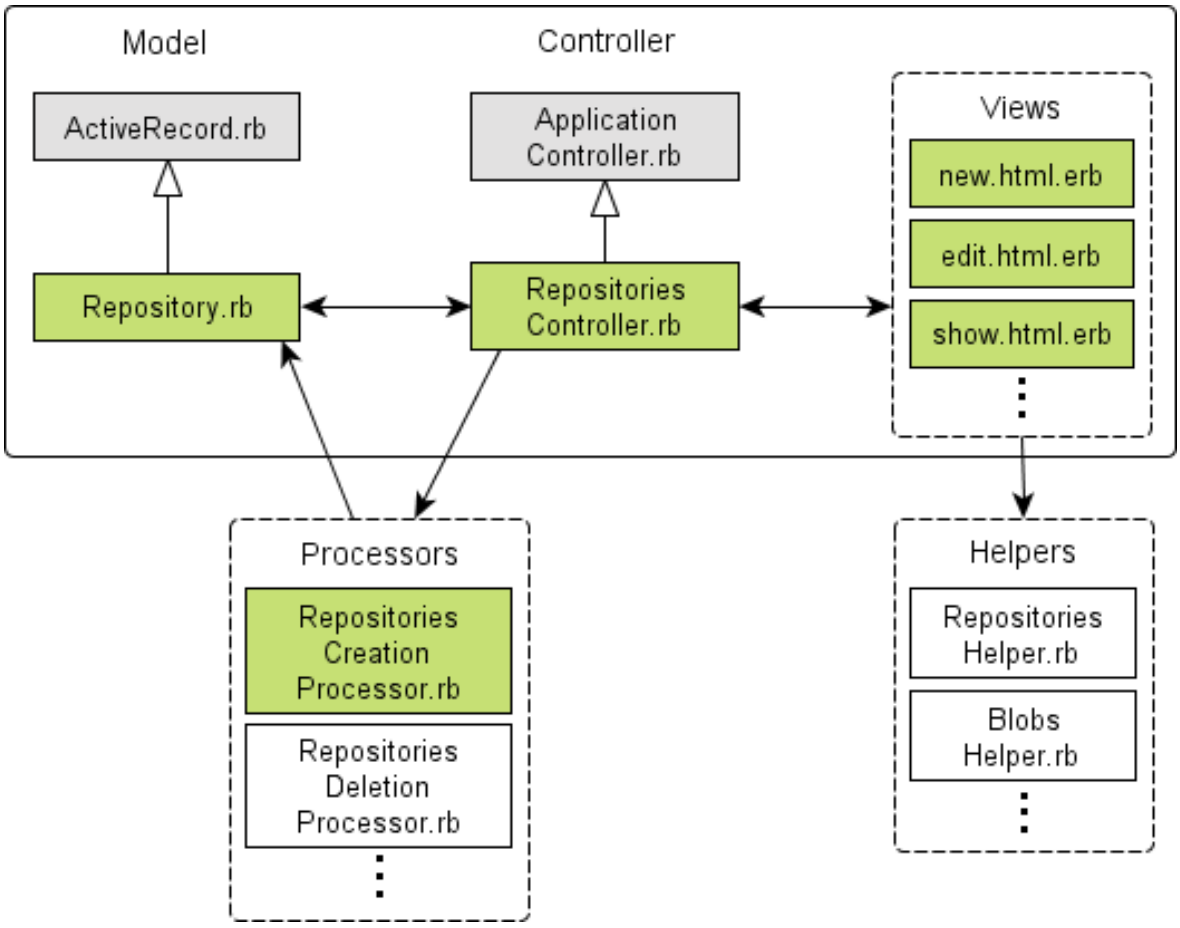

Figure 5.2: The class structure of `Repository` class.

Not all models were modified as extensively as `Repository` model. From the MVC architecture point of view, most of the modifications were done to the views as usability improvements. Figures 8.1 and 8.2 in Section 8.1 show the most essential

modifications done to Gitorious in Verso project.

5.4 Metafiles

Verso project developed to YouSource **two metafiles for each repository**. The metafiles describe information stored to the repository. The metafiles can describe the intented use, the content, the usage rights and the users of the repository. In Verso project however, only three pieces of information were set in the metafiles. A YML file holds information about the repository name and the possibility to allow merge requests. Another plain text file holds the repository description. The description was placed in its own file because it is the longest repository attribute and using the YML format with it was considered impractical.

The YML format of the file `repository.yml` is as follows

```
---
name: verso-project
merge_requests_enabled: true
```

A repository description metafile with the file name `description` can be a following one

```
This repository contains the modified Gitorious that's used
to run You\-Source.
```

The metadata of a project can't be updated via Git because projects are not Git repositories. The projects' attributes can only be changed through the WWW interface which makes the changes to the database. As the metadata of projects is saved to the database, projects don't have metafiles like repositories do.

6 Programming Practices

The programming in Verso project started after Gitorious was chosen as a platform for the development. From the start of the programming, the goal was to develop code that fits Ruby standards and is similar to the rest of the Gitorious code. During the project, the code was reviewed two times by the technical instructor Antti-Juhani Kaijanaho, who gave feedback on the developed code. Any questionable parts of the code were improved according to Kaijanaho's advice.

6.1 Formatting, Naming and Commenting Practices

Readability was the main guiding principle while formatting, naming and commenting the source code. Verso group followed their own defined **formatting** practices which were influenced by the original Gitorious source code. Two whitespaces were used as an indentation between code blocks. Only one command was written per line. Long lines were split to several lines using intendation to emphasize the continuity of the command.

When possible, conditional **commands** were written using positive logic, such as the statement `unless` of Ruby. SQL statements were made simpler by using the function `find` of `ActiveRecord` class of Ruby on Rails.

The **naming** of methods and variables was kept as self-explanatory as possible. The idea was to use as much self-explanatory names as possible so only a few clarifying comments were needed. Multipart names were written with the underline character between the parts. The class names were an exception and multipart class names were written together with every part starting with a capital letter. The constants were written with all capital letters. All the names in the code were written in English.

Commenting was kept minimal. The aim was to keep the code tidy and readable so that as few comments as possible would be needed. However, when a more complicated code was needed, it was accompanied with a comment explaining the idea. Commenting was also used when the implemented solution was considered imperfect. These parts were marked with `FIXME` comments to help the future developers find the problematic areas. Comments were written in English.

6.2 Source Code Example

Below is an example of source code illustrating formatting, naming and commenting practices used in Verso project taken from `repositories_controller.rb` file.

```
# encoding: utf-8
#--
#   Copyright (C) 2010 Juho Nieminen
#                      <juho.m.a.nieminen@jyu.fi>
#   Copyright (C) 2010 Tero Hänninen
#                      <tero.j.hanninen@jyu.fi>
#   Copyright (C) 2010 Marko Peltola
#                      <marko@markopeltola.com>
#   Copyright (C) 2009 Nokia Corporation and/or its
#                      subsidiary(-ies)
#   Copyright (C) 2007, 2008 Johan Sørensen
#                           <johan@johansorensen.com>
#   Copyright (C) 2008 David A. Cuadrado
#                      <krawek@gmail.com>
#   Copyright (C) 2008 Tor Arne Vestbø
#                      <tavestbo@trolltech.com>
#   Copyright (C) 2009 Fabio Akita
#                      <fabio.akita@gmail.com>
#
#   This program is free software: you can redistribute it
#   and/or modify it under the terms of the GNU Affero
#   General Public License as published by the Free Software
#   Foundation, either version 3 of the License, or (at
#   your option) any later version.
#
#   This program is distributed in the hope that it will be
#   useful, but WITHOUT ANY WARRANTY; without even the
#   implied warranty of MERCHANTABILITY or FITNESS FOR A
#   PARTICULAR PURPOSE. See the GNU Affero General Public
#   License for more details.
#
```

```
#   You should have received a copy of the GNU Affero
#   General Public License along with this program.
#   If not, see <http://www.gnu.org/licenses/>.
#++


def update_repo_with_zip
  @repository =
     @owner.repositories.find_by_name_in_project!(
                           params[:id],
                           @containing_project
                         )

  @root = Breadcrumb::UpdateRepositoryWithZip.new(@repository)
  target_head = ""
  if params[:target_head_selector] == ""
    target_head = params[:target_head]
  else
    target_head = params[:target_head_selector]
  end

  if !params[:local_package_file].blank?
    package_file = params[:local_package_file]

    temp_dir  = Repository.dir_for_temp_zip(
                           @repository.real_gitdir
                         )
    `mkdir #{temp_dir}`
    file_path = File.join(temp_dir,
                          sanitize_filename(
                            package_file.original_filename)
                          )

    File.open(file_path, "wb") {
      |f| f.write(package_file.read)
    }
```

```
      Repository.update_contents_from_zip(
                  @repository.real_gitdir,
                  file_path,
                  {"source_type" => "local_file",
                  "target_branch" => target_head}
                )

      #@repository.project.create_push_event(
      #                      @repository,
      #                      target_head,
      #                      current_user
      #                    ) # FIXME

      redirect_to [@repository.project_or_owner, @repository]

    elsif !params[:package_url].blank?

      Repository.update_contents_from_zip(
                  @repository.real_gitdir,
                  params[:package_url],
                  {"source_type" => "url",
                   "target_branch" => target_head}
                )

      #@repository.project.create_push_event(
      #                      @repository,
      #                      target_head,
      #                      current_user
      #                    ) # FIXME

      redirect_to [@repository.project_or_owner, @repository]

    else
      render :action => "update_repo_with_zip_form"
    end
  end
```

6.3 Grouping Practices

The source code was divided into different files according to the MVC architecture and Ruby on Rails practices. For example, the amount of embedded Ruby code was kept minimal in the view files by processing the information as much as possible in the corresponding controllers. Likewise, the controllers were kept clean by placing most of the data processing in the model files. Also, the appearance of the web pages was developed with a separate style sheet (CSS) file so that the HTML code would stay simple.

6.4 Development Platform

No single developing platform was used during the project. Each member of the project used a **text editor** of his choice to edit the source code. Gedit, Vim and Notepad++ were the mostly used editors.

The source code uses the UTF-8 character encoding. **Version control** was handled with Git and YouSource itself making the project self hosting which was one of the initial goals.

The **browsers** Firefox 3.5.9 and Konqueror 4.4.2 were used to test the user interface and the general functionality of the application. Firebug 1.5.3 plug-in was used to **debug** Javascript and CSS. The Ruby on Rails console was used to debug new features.

7 Testing Practices and Results

The developed YouSource software was tested during the project by Verso group, the customers, the instructors, about twelve initial users and two usability test users. The customers started to use the software as soon as it was set up in the test server. At that point no modifications were made to Gitorious. The initial users were invited to use the software after the second phase of the development when the first key features were implemented. The usability tests took place at the third phase of the development. In total, there were five phases of development.

Verso group tested YouSource constantly during the development. The application was used for maintaining and keeping the version history of its own source code. In other words, YouSource was in active and practical use by Verso group during the whole project. This practice revealed several bugs and usability issues from the application and also produced new ideas for future development.

All the testing practices used proved to be useful since a lot of errors were discovered. Daily use of the application brought up usability issues and functionality flaws, the integration testing of new features was crucial in debugging and the usability testing revealed several usability issues in the application as well as brought up new feature ideas to improve the usability. A couple of errors were also revealed in the system testing sessions that tested the overall performance of the application.

7.1 Integration Testing Practices

When a new feature was developed, it was instantly tested by its developer on his **work station** by putting the feature on its place and testing it in practice. After the first integration tests were carried out succesfully, a functional feature was then installed on the **test server** which was accessible for the group of initial users and the project organisation. Then the feature was tested again by its developer and at least one other project member to ensure that it worked well on the test server too.

The tests carried out were based on the functional requirements specified by Verso group. The functional requirements in turn were based on the needs and the goals described by the customer. The new features were implemented and tested one at a time. The most essential implemented functionalities can be seen in Figures 8.1 and 8.2 in Section 8.1.

No automatic testing was used, although the Ruby on Rails environment supports it. It was considered too time consuming to learn the testing side of Rails when the whole Ruby on Rails itself was a bit of a mystery.

7.2 Usability Testing Practices

YouSource's usability was tested in two usability testing sessions reported in [8] and [9]. Before the sessions not much had been done to improve the usability of the application. The focus of development had been on implementing the required new features. Still, as one of the main ideas behind the project was to encourage more people to use version control, it was necessary to put effort into usability too.

The **usability sessions** were carried out by two testers from Verso group and a user who performed the defined test cases using a laptop with Windows XP operating system. In the start, the testers asked questions about the background of the user. Then the user was told to release something on YouSource website. To be more specific, the test cases were logging in, uploading an SSH key, creating a project, creating a repository and uploading files to the repository. The user was instructed to think aloud so that the testers could know what the user was thinking and what the user was going to do next. The releasing was conducted via Git, and for that the user was instructed to use Putty to connect to a server called `charra.it.jyu.fi` with Git installed.

The both usability testing sessions were conducted very much the same way but the skills of the **users were very different**. The first user was an active programmer and was used to operate with command line interface and version control. The second user instead was less experienced and not very familiar with SSH keys, Git or Linux bash. This diversity was good for the usability testing because the sessions brought up different issues.

The usability of the application was also put to test in the system testing sessions where all the main functions of the application were tested. More about system testing is described in Section 7.3.

7.3 System Testing Practices

The functionalities of the application were tested in two system testing sessions performed by Tero Hänninen. System Testing Plan [3] describes the tests that were carried out in the sessions and System Testing Reports [4] and [5] discuss the results of the sessions.

The system **test cases** included all the features that were developed during Verso project as well as the most essential features provided directly by Gitorious like creating a project and a repository. Every test has a set preconditions, test steps and postconditions that all must pass for the test to pass. The testing reports contain a list of the performed test cases, the results of each test case and a comment describing the failures or the observations.

7.4 Testing Results

The **daily use and testing** of the application by Verso group revealed several issues, some of which were discovered by accident. The use also brought up improvement ideas for already implemented features. One example of a **feature that got redesigned** based on improvement ideas was the possibility to create a project and a repository simultaneously so that new users would get started with the application quicker. The feature was implemented and remained untouched for some time until problems in error handling started to show up which led to separation of the project and repository creation. Section 8.4 explains the issue more.

An example of a problem that was **discovered by accident** during the daily use was the deletion of a user or a team avatar. The feature was a part of Gitorious and it was not required by the customer. However, it was discovered, that the deletion of avatar images didn't work properly. The problem was solved and the fix sent back to Gitorious where it got accepted as a fix. Without an extensive use of the application these kind of problems that where out of the required features might have gone unnoticed.

Besides Verso group the instructors and the customer's representatives used YouSource during the development. Their practical usability testing of the software brought up many issues related both to the features that were being developed by Verso group and to the features already implemented in Gitorious. The instructors

and the customer's representatives supplied their testing results mostly straight to Trac which helped to keep the feedback organized. For example, one of the problems that were discovered this way was the issue that without a master branch the *source tree* and *commit log* pages of a repository failed to load.

The **unit and integration testing** of new features was very useful in finding errors. For example, the visibility of projects and repositories was thoroughly tested because the more tests were carried out, the more places were discovered that wrongly displayed private projects or repositories. Another good example is the feature of adding a single file to a repository. While testing, it wasn't always clear to the developer what the behavior of Git would be in certain situations. For instance, when a file was added to a former version of the repository, the head pointer not being at the latest commit, the results were unexpected. It turned out that Git makes a new branch for the added file starting from the former version, i.e. the commit where the head pointer was.

The **usability testing sessions** brought up a number of usability issues. Some of the problems were detected on both sessions and some came up only on either one of the two. Problems with *SSH key help* and *Getting started message* were noted on both sessions. Other ideas that came up during the sessions were a feature to compare two different commits, a feature to ask the user for confirmation before he is added to a team, and a page where a user can get information about how to get started with the application.

In the first **system testing** session two errors were discovered. One concerned private projects and their visibility. If a project was marked to be visible only to the users logged in, it still showed the project in search results even if the current user wasn't logged in. The other error was related to mirroring an SVN repository. It turned out that creating an SVN mirror repository doesn't work at all on the test server. The feature was tested again in the work stations and it worked fine. It remained unknown why it failed on the test server. The problem was agreed with the customer to be handled in future development.

The instructor **Jukka-Pekka Santanen** tested the WWW interface of YouSource and discovered several usability issues. Some of them are described in Section 4.12. His testing brought up many good improvements and not all of them could be implemented because of lack of time, but they were agreed with the customer for future development. In addition, one bug was found as a result of his testing. It concerned project URL attributes (home URL, mailinglist URL and bugtracker URL),

and caused a crash when the URL was badly formed. This was solved with better URL validation and error handling. The fix was also offered to Gitorious and it was approved.

8 Realization of Objectives

The chapter describes the requirements set to YouSource and how well they were achieved. The main features that were developed by Verso group were login using KorppiLDAP, private projects and repositories, a repository browser page and different updating and mirroring methods for repositories. The usability modifications are discussed in Section 4.12. The features that were agreed with the customer for future development include possibility to comment and certificate source code, a button for automatic merging, editing of text based files via the WWW interface and requiring the user's SSH key only when truly needed.

It should be noted, that Verso project initially set out to develop a system that enables releasing of source code among a work community. This requirement was not fullfilled in the true meaning of the term release a piece of source code. In Yousource a **user can't release a single version** of his source code. Instead, if he has decided to share his work on the web site, all the version history of the source code is accessible. However, the customer's representatives made it clear from the start, that a system with version control build in was one of the main requirements. Furthermore, the customer did not actually demand a system with a clear single version releasing feature. Still, the need for such a feature has been noted in Section 8.5.

8.1 Realization of Requirements

All eleven mandatory and four out of six important **functional requirements** of the application described in [2] were fulfilled in the project. Possibility to authenticate users with Kerberos and requiring the user's SSH key only when truly needed were considered important but were agreed with the customer to further development because of lack of time. Also four out of thirteen of the features marked as useful were implemented.

Most of the **usability improvement ideas** that came up during the project were implemented to YouSource. The ones that were agreed to further development were considered too time consuming or difficult, or they came up too late so there was no time left to implement them in the project. Usability modifications are described in Section 4.12.

Figures 8.1 and 8.2 present simplified views of Gitorious before and after Verso

project. Figure 8.1 shows the project orientation of Gitorious that was diminished in YouSource as shown in Figure 8.2. All the main functionalities that were developed in Verso project are visible in Figure 8.2.

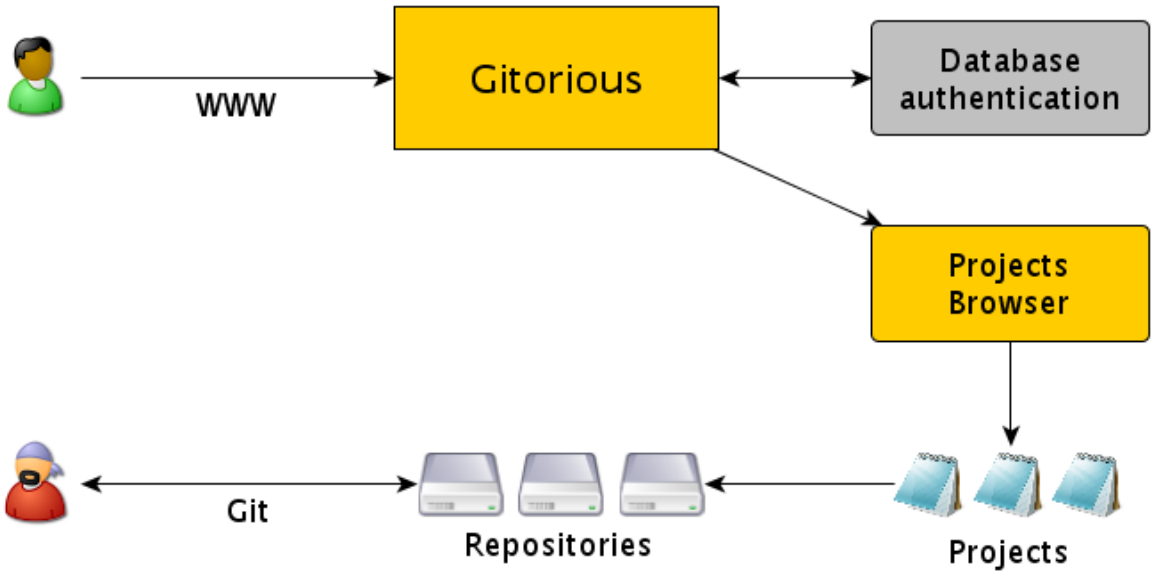

Figure 8.1: Simplified structure of Gitorious at the start of Verso project.

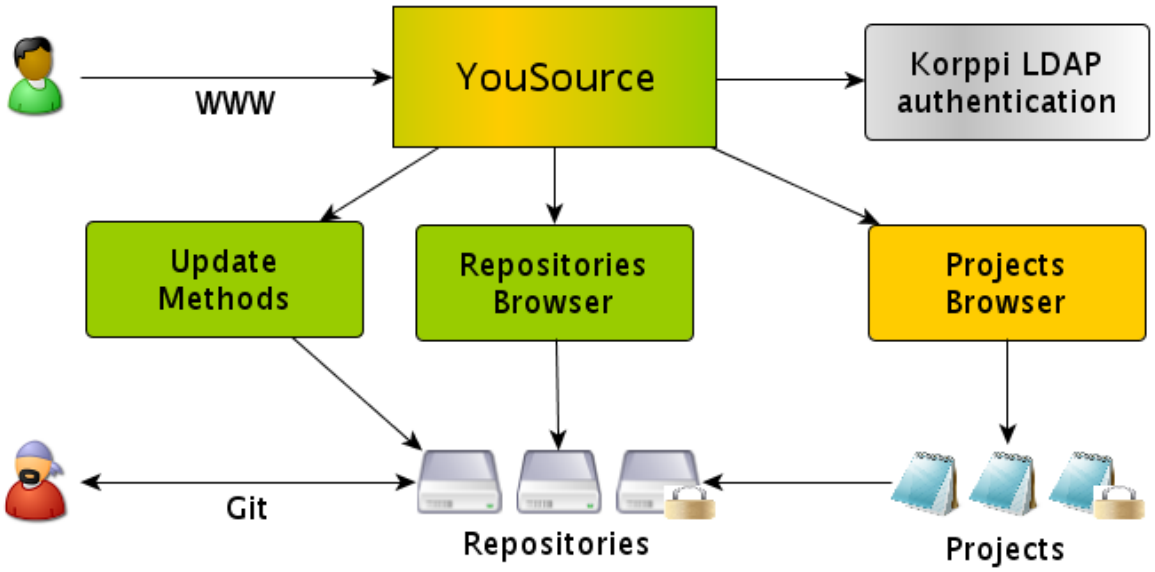

Figure 8.2: Simplified structure of YouSource at the end of Verso project.

8.2 Unsatisfactory Solutions in Implementation

The **SVN mirroring for a repository** was a requested feature from the customer. There was a need to share source code on YouSource also for the users who use SVN version control. The feature was developed and tested locally on the work stations and then pushed to the server. Even though the feature worked fine locally, it does not seem to work on the web server. The reason for this is currently unknown.

The **URLs** that a user can give to zip and SVN mirroring and to zip updating are **validated but not checked** whether they respond or not. The validation should be expanded so that it checks that the provided URL really contains a zip package or an SVN repository. In a case of an error, a notification should be shown to the user. The handling of these exceptions was not implemented during the project because of lack of time and expertise.

A helper module was developed to **update the metafile** information to the the database. However, as the technical instructor Antti-Juhani Kaijanaho pointed out, the metafile helper `GmrfHelper` is not actually a helper class in the sense of Ruby on Rails because helpers are normally classes related to the views that aid to display data to the user. The metafile helper should be implemented to `repository` module.

Yet another poor implementation is *Repositories* page, also known as the **repository browser** (see Figure 4.7 in Section 4.6). The feature works nicely, but the source code for it is in a wrong controller. One might think that a page called repositories is situated in the repositories controller, but instead, it currently resides in the site controller. It was placed there because Verso group didn't have enough knowledge on the routing in Ruby on Rails. A proper way would have been to implement the routing through the repositories controller. Gitorious developers had made the repositories routing in a complicated way and Verso group didn't know how to modify the repository routes needed for the repository browser.

The last paragraph in Section 4.3 describes an issue with the **terms of service**, an example of another unsatisfactory solution in the application.

8.3 Challenges in Implementation

Most of the problems faced in development of YouSource were related to the **test server**. Some features behaved differently on the work stations compared to the test server, like the SVN mirroring (see Section 8.2). On the other hand, KorppiLDAP logging feature didn't work at all on the work stations but on the test server no problems were encountered. Therefore, some bugs were not spotted until they were out for the users to find. Luckily, Gitorious provided a feature that let all the errors happening on the test server to be emailed to Verso group. The feature enabled a quick response to errors and helped to discover errors that were hard to find.

Gitorious sometimes works unreliably when the **application is restarted**. In some cases Poller daemon won't shut down in the restarting process but instead another Poller deamon is created which results in two Pollers running at the same time. Until the issue was discovered, there were half a dozen hours lost because, for example, creating repositories didn't work properly if more than one Pollers were running.

A new working tool Git, a new programming language Ruby and a new programming framework Ruby on Rails were also a challenge in Verso project. The members of **Verso group had no experience** on any of them before the project. However, all these turned out to be very manageable tools and learning them was not a big issue. Possibly due to the tools, Verso group was able to achieve a good work flow that might not have been possible with more inflexible set of tools.

8.4 Modifications during the Implementation

Creating a project and a repository on a single form was changed to be handled in two separate forms. The change was done after it was discovered that it was very difficult to implement error handling into a single form that is handling two different objects. At first, the feature had only one form which asked information for a project and a repository and then created them both simultaneously. After discovering that the page would crash on some situations, the feature was then changed so that it was no longer possible to create a project and a repository at the same time. Instead, if a user with no projects comes to *Create a new repository* page (see Figure 4.4 in Section 4.4), a new function provides a link to *Create a new project* page. Next to the link there is a button to create a generic project with one click. If the user chooses the generic project, the application creates it and returns to the creation of a

repository, which was the intention of the user in the first place.

In the second source code review the tecnical instructor Antti-Juhani Kaijanaho pointed out a couple of imperfect implementations. First, the use of **command line in Ruby code was insecure** as it allowed a user to inject malicious code. The commands using the backtick ` method were replaced with `exec` commands which perform the same tasks more safely. Second, when passing the username and password to the script `ldap_authenticate.py`, once again the backtick method was used. Here, a new problem was found as the password was visible to the users of the operating system for a brief moment when the process was initiated. The backtick method was replaced with `IO.popen` method to prevent the revealing of the passwords. Third, the LDAP authentication script was modified so that no malicious code can be injected while printing into the log. However, none of the changes were visible to the user.

8.5 Further Development Ideas

New ideas for features, usability improvements and bug corrections were suggested by the customer, the instructors, the usability test users, the usability expert Meeri Mäntylä and by the group members. Most of these ideas were agreed with the customer for future developers because of lack of time in the project. A full list of the development ideas to improve various parts of the application can be found from the requirement specification [2].

User authentication with Kerberos would enable users from other universities to log in to YouSource and collaborative projects to gain members more easily.

SSH should be required only when truly needed. Alternative update methods (e.g. zip update) to Git were developed to YouSource. Some people might be using the application solely with those methods, so requiring a SSH key before a user can create a project is unnecessary.

Text based files could be edited through the WWW interface. YouSource has wiki pages which are actually stored in a Git repository. Those pages can be modified on the website and their version history is available too. The feature could be expanded to all text based files in all repositories. Furthermore, for source code files a javasript based programming API (like Bespin [7]) could be helpful.

The right to certificate source code files and commits could be given to certain

users. The certificate would be a sign to other users that the piece of code is of good quality and it does what it is supposed to.

The possibility to accept merge requests on YouSource website could be added. YouSource notifies the user if there is a merge request for any of his repositories, but the merges can only be made with Git. Instead, the application could try to do the merge and perform it if no conflicts are detected. Otherwise the merge would have to be done manually.

Releasing of static files on Yousource could be implemented. This means that the users could release files that have no version history. This would be useful if one wants to release a single version of his software. It might also be easier for some users to handle static files because then it would always be clear what file they need to download in order to test the latest version of the software. In Git this is done with tags by marking a commit with a release tag.

9 Guide for Future Developers

The customers from the Department of Mathematical Information Technology decided that the software developed in Verso project is going to be further developed after Verso project ends. The chapter provides some tips to guide the future developers so that the most essential issues of the software would be addressed as soon as possible. For a more extensive look on the open issues, please refer to Verso Trac [2].

9.1 Essential Bugs

YouSource currently has nine known bugs and some of them limit the usability of the software considerably. The following bugs should be highly prioritized when the further development is started.

1. SVN mirroring is not working on the test server (see Section 8.2).
2. When a team member is removed, he can still create a repository under the projects owned by the team.
3. *Activities* page (see Figure 9.1) crashes if an active project is removed.

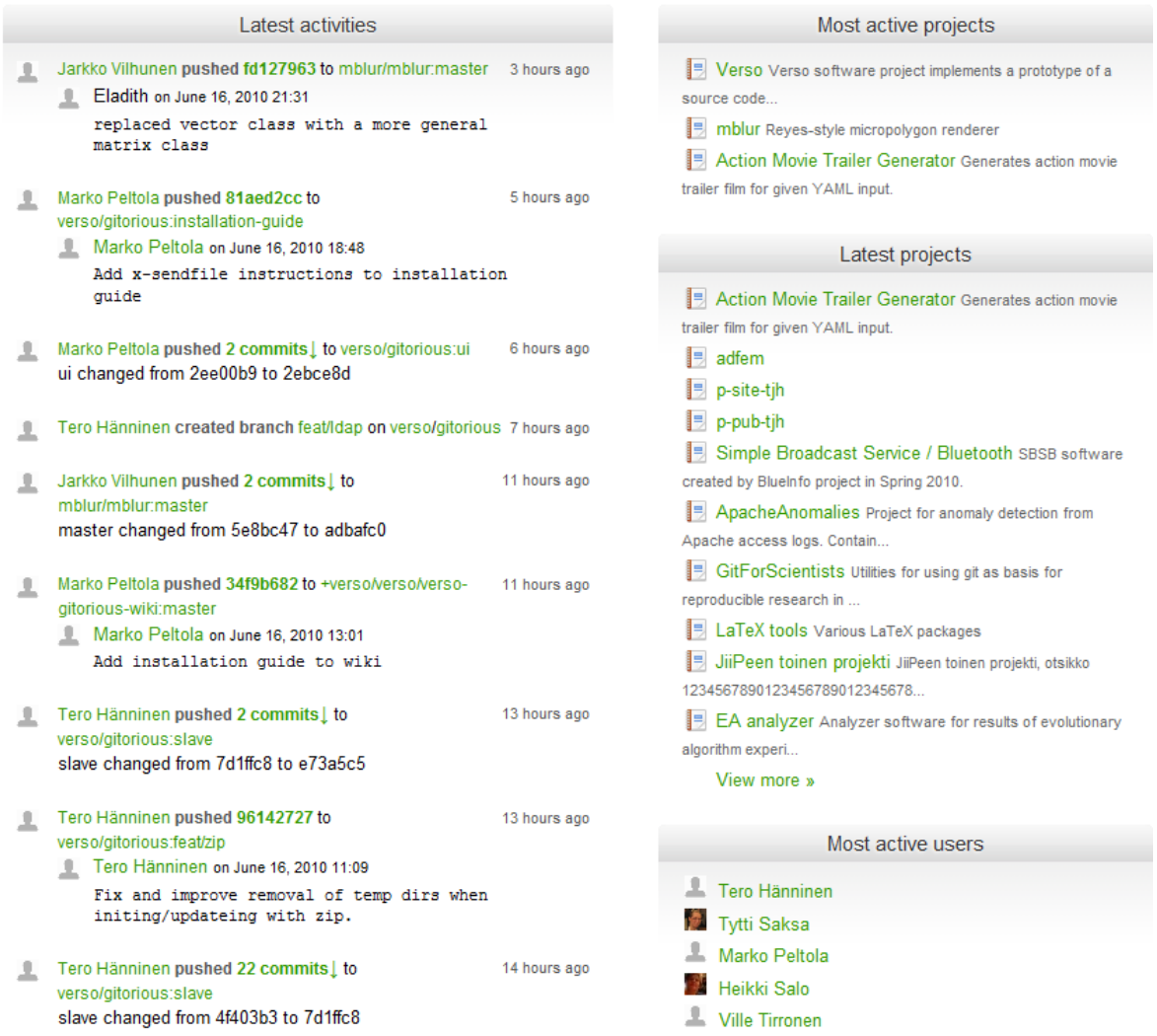

Figure 9.1: *Activities* page of YouSource.

9.2 Improvements to Existing Features

Before starting to improving the existing features the developers should go through the unsatisfactory implementation solutions described in Section 8.2. In addition, the following features should be taken into account.

1. When updating a repository with a zip package a possibility to add a commit message should be implemented.
2. Uploading of the user's first SSH key should be added as part of the project or repository creation process.

3. The *Getting started message* should be improved to include more information about Git usage such as the command `git init` if the command `git clone` fails for some reason.

9.3 The Most Useful New Features

Section 8.5 describes some of the feature ideas that came up during Verso project and were discussed in the project meetings. Those features should take priority when planning for new feature implementations.

In addition, the following feature ideas should be considered as the most useful during the further development.

1. A how-to-get-started page which describes the most essential features of the application and points the user to the right direction should be added.
2. A missing license of a repository should be highlighted.
3. Sorting and searching functions should be implemented to *Repositories* (see Figure 4.7 in Section 4.6), *Projects* and *Teams* pages.
4. A possibility to add keywords to repositories should be implemented.
5. All new and existing users should be asked to accept the terms of service.

9.4 Development Practices

Section 14.1 in project report [10] describes some practices that were found ineffective during the project or would have been useful if they had been used. Two important things that should have been done in Verso project were found.

Make use of the branching of Git. Every little new feature and improvement should be done in its own branch and not in the master branch. This practice will help to keep the version history clean and merge requests to upstream are easier to do.

Take full advantage of the ticketing system. Tickets should have a clear naming and tagging practice so that there is never confusion about what should be done next or what the priorities are. It also makes browsing the tickets much easier.

10 Summary

Verso project developed a source code management and releasing software called YouSource for Department of Mathematical Information Technology in University of Jyväskylä. YouSource will be used for testing the practices of source code sharing and version control usage among the employees of the department. YouSource is a web application developed with Ruby on Rails and based on the open source software called Gitorious.

On Gitorious users can host their Git repositories, create projects and teams, manage merge requests of their repositories and read activity feeds of everything happening on the website. Verso project developed half a dozen new features to Gitorious such as private projects and repositories, alternative update methods and user authentication using KorppiLDAP interface. In addition, the usability of the application was improved considerably as well as the appearance of the website was changed with a new logo and a color theme.

During the project, YouSource was in daily active use by Verso group, the customer and a couple of test users. This was possible because, as being a source code management website, YouSource could be used to host and manage its own source code. Besides active use, YouSource was tested in two usability testing sessions, feature integration tests and in two system testing sessions. The application was also once reviewed by a usability expert and the source code was reviewed twice by the technical instructor of Verso project.

Verso project managed to complete all of the essential requirements set for the prototype. Moreover, the customer decided to continue the development of YouSource after the project. This means that the future developers can utilize the many development ideas for the application that came up during Verso project.

11 References

[1] GitHub Inc., *Secure Source Code Hosting and Collaborative Development*, available at URL: <http://github.com>, 2010.

[2] Tero Hänninen, Juho Nieminen, Marko Peltola and Heikki Salo, *Verso Project, Requirement Specification in Trac*, University of Jyväskylä, Department of Mathematical Information Technology, available at URL: <https://trac.cc.jyu.fi/projects/verso/>, 2010.

[3] Tero Hänninen, *Verso Project, System Testing Plan*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.

[4] Tero Hänninen, *Verso Project, System Testing Report, RC-1*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.

[5] Tero Hänninen, *Verso Project, System Testing Report, RC-2*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.

[6] Tero Hänninen, *Verso Project, Comparison of Platforms*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.

[7] Mozilla, *Bespin*, *Mozilla Labs*, URL: <http://mozillalabs.com/bespin/>, 2010.

[8] Juho Nieminen, *Verso Project, Muistio, 1. käytettävyystestaus*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.

[9] Juho Nieminen, *Verso Project, Muistio, 2. käytettävyystestaus*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.

[10] Heikki Salo, *Verso Project, Projektiraportti*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.

[11] Johan Sørensen, *About Gitorious*, available at URL: <http://gitorious.org/about>, Shortcut AS, 2009.

[12] Sanna Talja, *Information Sharing in Academic Communities: Types and Levels of Collaboration in Information Seeking and Use*, New Review of Information Behavior Research, vol. 3, pages 143–160, available at URL:

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=-10.1.1.96.163&rep=rep1&type=pdf>, Taylor Graham Publishing, 2002.