

**Verso Project**

**Application Report**

**Tero Hänninen**  
**Juho Nieminen**  
**Marko Peltola**  
**Heikki Salo**

Version 0.4  
Public  
3.6.2010

**University of Jyväskylä**  
**Department of Mathematical Information Technology**  
**Jyväskylä**

<b>Acceptor</b>	<b>Date</b>	<b>Signature</b>	<b>Clarification</b>
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:** 40

**Abstract:** This document describes the software that was developed in Verso Project, the realization of objectives concerning the software and the programming and testing practises in the project.

**Keywords:** Git, Gitorious, Ruby on Rails, source code management, YouSource, web application.

## 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/](https://korppi.jyu.fi/kotka/servlet/list-archive/verso/) and  
[https://korppi.jyu.fi/kotka/servlet/  
list-archive/yousource-users.group/](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 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 user interface chapter. The background chapter was modified. The sitemap and metafiles subsections 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 header and sidebar were merged into the chapter page structure. The figure of inner and external interfaces was added and content for its chapter was written. The subsections system testing and unsatisfactory implementations were written.	JN
0.3	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
0.3.1	26.5.2010	The references were modified. Testing chapter was modified and expanded with testing results section. The realization of objectives chapter was continued.	JN

0.3.2	27.5.2010	The chapter realization of objectives and the section testing results were continued.	JN
0.3.3	28.5.2010	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 guide for future developers was written. Section logging in was extended. The figures of YouSource before and after Verso project were added.	JN
0.3.5	1.6.2010	The chapters introduction, terminology, background and application structure were modified.	JN
0.4	2.6.2010	The chapters application structure, user interface, testing practices, programming practices and realization of objectives were modified. The summary was started.	JN





# Contents

<b>Project Contact Information</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Terminology</b>	<b>2</b>
2.1 General Terms . . . . .	2
2.2 Verso Specific Terms . . . . .	2
<b>3 Background</b>	<b>4</b>
3.1 Publishing Channel for Source Codes . . . . .	4
3.2 Gitorious as a Starting Point . . . . .	4
<b>4 User Interface</b>	<b>6</b>
4.1 Sitemap . . . . .	6
4.2 Page Structure . . . . .	7
4.3 Logging in . . . . .	8
4.4 Creating a Repository . . . . .	9
4.5 Creating a Mirror Repository . . . . .	10
4.6 Repository Browser . . . . .	10
4.7 Updating Repository with a Zip Package . . . . .	12
4.8 Adding Single Files to a Repository . . . . .	13
4.9 Private Repository . . . . .	13
4.10 Private Project . . . . .	14
4.11 Editing of Repository Metafiles . . . . .	14
4.12 Usability Modifications to the WWW Interface . . . . .	15
<b>5 Application Structure</b>	<b>18</b>
5.1 Inner and External Interfaces . . . . .	18
5.2 Components . . . . .	19
5.3 Class Structure . . . . .	20
5.4 Metafiles . . . . .	20
<b>6 Programming Practices</b>	<b>22</b>
6.1 Formatting, Naming and Commenting Practices . . . . .	22
6.2 Grouping Practices . . . . .	23
6.3 Development Platform . . . . .	23

---

6.4	Source Code Example . . . . .	23
<b>7</b>	<b>Testing</b>	<b>27</b>
7.1	Integration Testing . . . . .	27
7.2	Usability Testing . . . . .	27
7.3	System Testing . . . . .	28
7.4	Testing Results . . . . .	28
<b>8</b>	<b>Realization of Objectives</b>	<b>31</b>
8.1	Realization of Requirements . . . . .	31
8.2	Unsatisfactory Solutions in Implementations . . . . .	32
8.3	Complications in Implementation . . . . .	33
8.4	Modifications in Implementation during the Project . . . . .	34
8.5	Further Development Ideas . . . . .	34
<b>9</b>	<b>Guide for Future Developers</b>	<b>36</b>
9.1	Important Bugs . . . . .	36
9.2	Improvements to Existing Features . . . . .	36
9.3	The Most Useful New Features . . . . .	37
9.4	Advice for Development . . . . .	37
<b>10</b>	<b>Summary</b>	<b>38</b>
<b>11</b>	<b>References</b>	<b>39</b>

# 1 Introduction

In a work community, sharing information increases productivity [10]. For this reason the Department of Mathematical Information Technology at University of Jyväskylä decided to start developing a method for researchers to share source code with each other. The idea was first tried out by Ville Tirronen who tested a prototype software that did not provide the functionalities needed by the users. A need for such a software was still recognized but no free or inexpensive solution was found from the market. The idea was then proposed for a student project that was starting in the MIT department. The idea was approved and the project, soon to be known as Verso, started to further develop the idea of source code publishing software.

Verso project 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 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.

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 tells more about the background of the project. Chapter 4 describes the user interface of the application concentrating on the parts that were 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 tells how testing was carried out. In Chapter 8 it is considered how the objectives set to the application were fulfilled and finally in Chapter 9 ideas on how to start the further development are presented.

Verso project wrote several documents describing the software and the practices of the project that support this document. The realization of the goals and practices is described in the project report [12]. The usability testing sessions are described in two memos [8] [9]. System testing is described in System Testing Plan [3] and System Test Results [?]. The comparison results between possible software platforms in the beginning of the project is described in Comparison of Platforms [6].

## 2 Terminology

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

### 2.1 General Terms

These 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.

<b>Branch</b>	in Git is a pointer to a commit. The current branch determines where the user's new commits will go.
<b>Commit</b>	contains file modification data and a log message produced by the user describing the changes.
<b>MVC architecture</b>	i.e. Model-View-Controller, is an architectural pattern used in software engineering. The pattern isolates the application logic from input and presentation, permitting independent development, testing and maintenance of each.
<b>Push</b>	uploads the new commits in the local repository to a remote repository. It can be thought as releasing a version of a code.
<b>Ruby on Rails</b>	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.

<b>Owner</b>	is a user or a team. An owner has 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 his projects' and repositories' details.
--------------	--

---

<b>Private project</b>	means that the project can only be seen by its members.
<b>Private repository</b>	means that the repository can only be seen by users that have view, review, commit or administrate rights to it.
<b>Project</b>	is a user created entity in the Verso system 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.
<b>Public</b>	means that a repository or a project is seen by all users.
<b>Repository</b>	refers to a Git repository which belongs to a project. It stores data and the version history of that data. It can have several attributes, for example a description and a license. A repository is owned by a user or a team.
<b>Team</b>	is a group of users. One or more users are team admins and the rest are team members.
<b>Viewer</b>	is a right that a user can have for a repository. It grants the ability to see the repository even if it is marked as private. The wiewer right is the lowest right of the possible rights to a repository. Others are in order review, commit and administrate.
<b>YouSource</b>	is the name of the application developed by the 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 of the developed software and 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 the Department of Mathematical Information Technology in University of Jyväskylä, there is no common system for version control. This causes a number of problems. For instance, when a worker leaves the department all the source code he has developed often leaves too. The lack of a single version control system also prevents workers from knowing who is doing what, which may lead to producing overlapping work. Furthermore, the current disorganized situation presents licensing problems in which one is unaware who owns a piece of source code and how can it be used.

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

### 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 [6] and finally the Verso group ended up with Gitorious [13].

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, GNU Savannah, Project Kenai, Fedora Hosted and KnowledgeForge. The disadvantages with these were inactivity, difficult repository creation process, no activity view for repositories and 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 main features 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 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. Most of the look of Gitorious was left untouched but almost all the new features needed a new user interface implementation. Not only the web pages related to the new features were changed but also minor changes were made here and there to improve usability. The chapter describes the user interface changes that were done to Gitorious by the Verso group.

### 4.1 Sitemap

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.

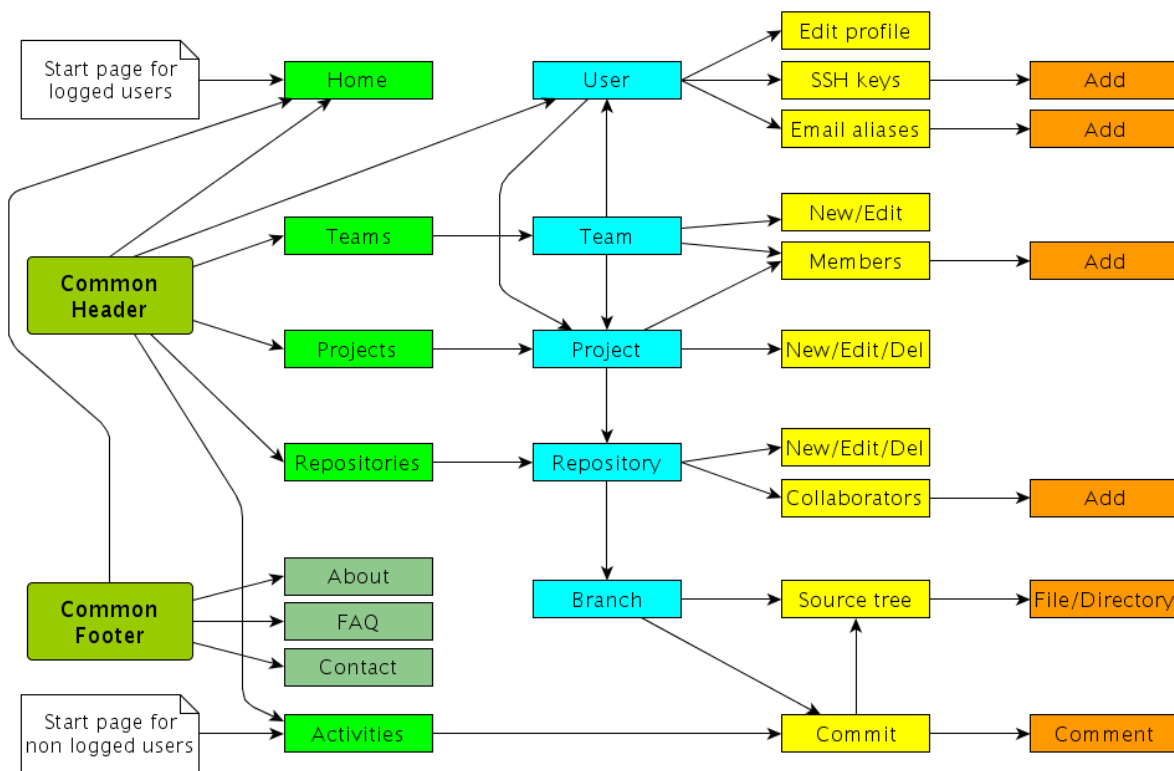


Figure 4.1: Simplified sitemap of YouSource.

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 vertical axis displays the logical dependencies between the pages. For instance, a project has many repositories and a branch has many commits.

## 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 control.

The screenshot displays the 'Source tree for verso-project' page. The header features the YouSource logo, navigation links for 'Activities', 'Projects', 'Repositories', and 'Teams', a search bar, and user information for 'Juho Nieminen' including 'Home', 'Profile', '0' notifications, and 'Logout'. The breadcrumb trail shows the path: 'Verso > verso-project > master > /'. The main content area contains a table of files with columns for 'File name', 'Last updated', and 'Latest commit message'. The files listed are 'application/' (updated 17 May 16:10), 'project/' (updated 18 May 13:00), and 'README' (updated 03 May 15:35). Below the table are input fields for 'Add file:' and 'with message:', along with 'Browse...' and 'Upload file' buttons. A note states: 'Note: you can also update files by uploading a file with the same name.' The sidebar on the right provides repository details: 'Repository: verso-project', 'Project: Verso', 'Owner: verso', 'Branch: master', 'HEAD: 23a6270', and 'HEAD tree: 6251f92'. It also includes links for 'Commit log' and 'Download master as tar.gz', and a list of branches: 'master', 'katselemointi2', 'projektiraportti\_0.2', and 'yoursource\_metafiles'. The footer contains navigation links: 'Home | About YouSource | FAQ | Contact' and the 'GITORIOUS' logo.

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

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 control. 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 root pages like the four main pages (*activities, projects, repositories, teams*) or project creation and team creation pages.

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

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 informative pages (about, FAQ, contact). A full view of a page in YouSource that displays all the elements described in this chapter is shown in Figure 4.2.

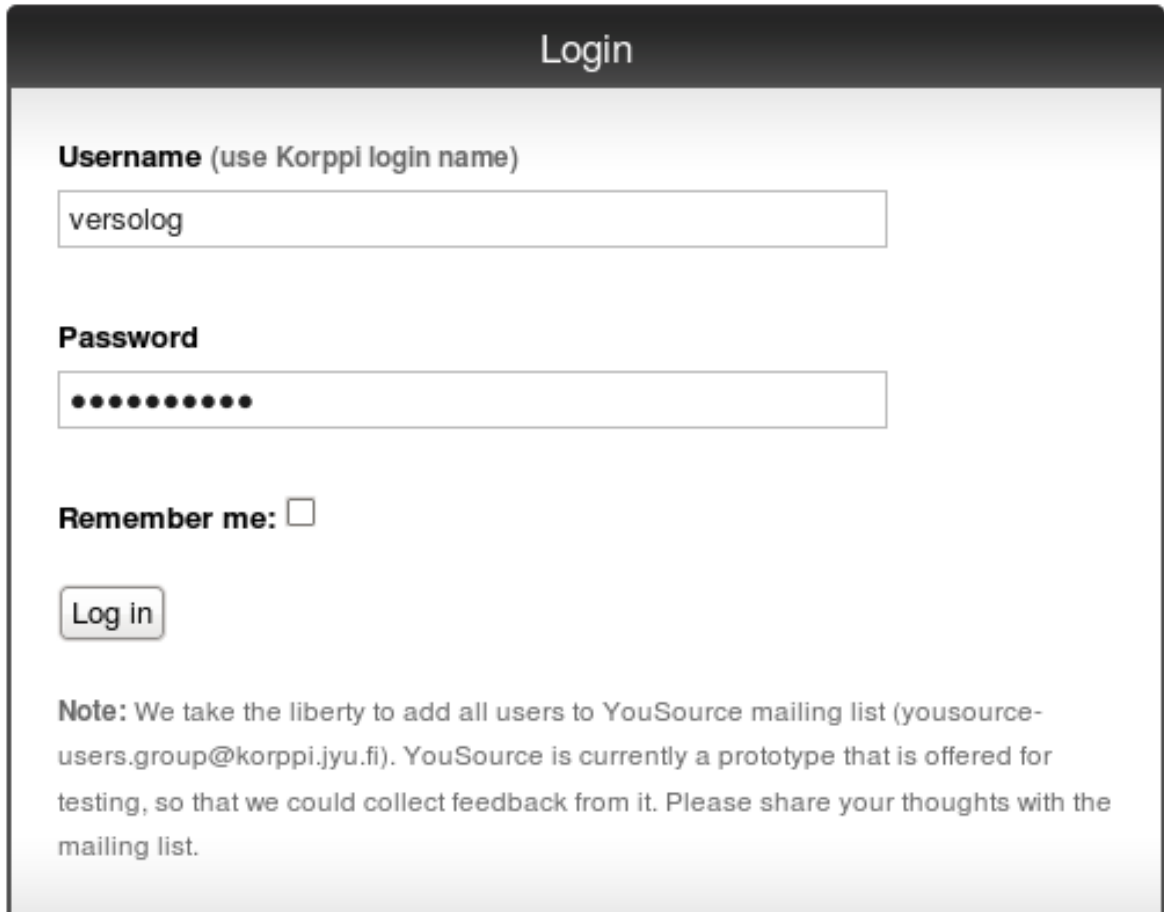
### 4.3 Logging in

Gitorious uses email for logging users into the system. This was considered inconsistent because all the current web applications in University of Jyväskylä use a username for logging in. Therefore, YouSource was developed to use the username too instead of email. Gitorious also 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.

One of the main application 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.

When users create a new account in Gitorious, they 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 option to use terms of service in the future was sustained, 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.



The screenshot shows a login form titled "Login". It contains the following elements:

- Username (use Korppi login name)**: A text input field containing the text "versolog".
- Password**: A password input field with ten dots representing the masked password.
- Remember me:** A checkbox that is currently unchecked.
- Log in**: A button with the text "Log in".
- Note:** A paragraph of text stating: "We take the liberty to add all users to YouSource mailing list (yousource-users.group@korppi.jyu.fi). YouSource is currently a prototype that is offered for testing, so that we could collect feedback from it. Please share your thoughts with the mailing list."

Figure 4.3: The login view.

## 4.4 Creating a Repository

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 a SVN repository. If the user prefers not to use any of these options, a normal empty repository will be created.

Figure 4.4 displays the form that is used to create a new repository. The form has two sections: basic information and options. In the basic information section the user specifies in which project the repository will be created and what name, description and license will it have. The description is an optional attribute. The 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.

## 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 zip will be added to it. On the other hand, if the user selects 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. A script takes care of the daily update. It will go through all the mirror repositories in YouSource and checks if there's any changes in the source file or repository at the mirror URL. The mirror URL can be changed at any time on the repository edit page which can be accessed through the sidebar of the 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 this page is one of the four main pages which are activities, projects, repositories and teams. All the main pages have a link in the common header in each page.

## Create a new repository

The repository will be owned by the user [Juho Nieminen](#)

### Basic information

**Add repository to project:**  (or start a [new project](#)).

**Repository name** (Allowed characters: A-Z, 0-9, \_, -):

**Description** (Use [Markdown](#) for formatting):

**License:**

### Options

**Initialize repository with a local zip-file** (optional):

The contents of the zip file will be automatically stored into the new repository.

**Mirroring settings:**  No mirroring  SVN mirroring  ZIP mirroring

**Mirror url:**

When mirroring is enabled, this repository will be automatically updated from an SVN repository or a ZIP file.

**Private repository:**

Only repository collaborators are allowed to see private repositories. You can add collaborators in Manage collaborators section.

**Enable merge requests:**

By choosing the check box you allow other YouSource users to send you requests to merge changes they have made to your code.

Figure 4.4: The form for creating a new repository.

The repositories page 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 paginated for twenty repositories which means 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 high-

lighted as is seen in Figure 4.5. 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.

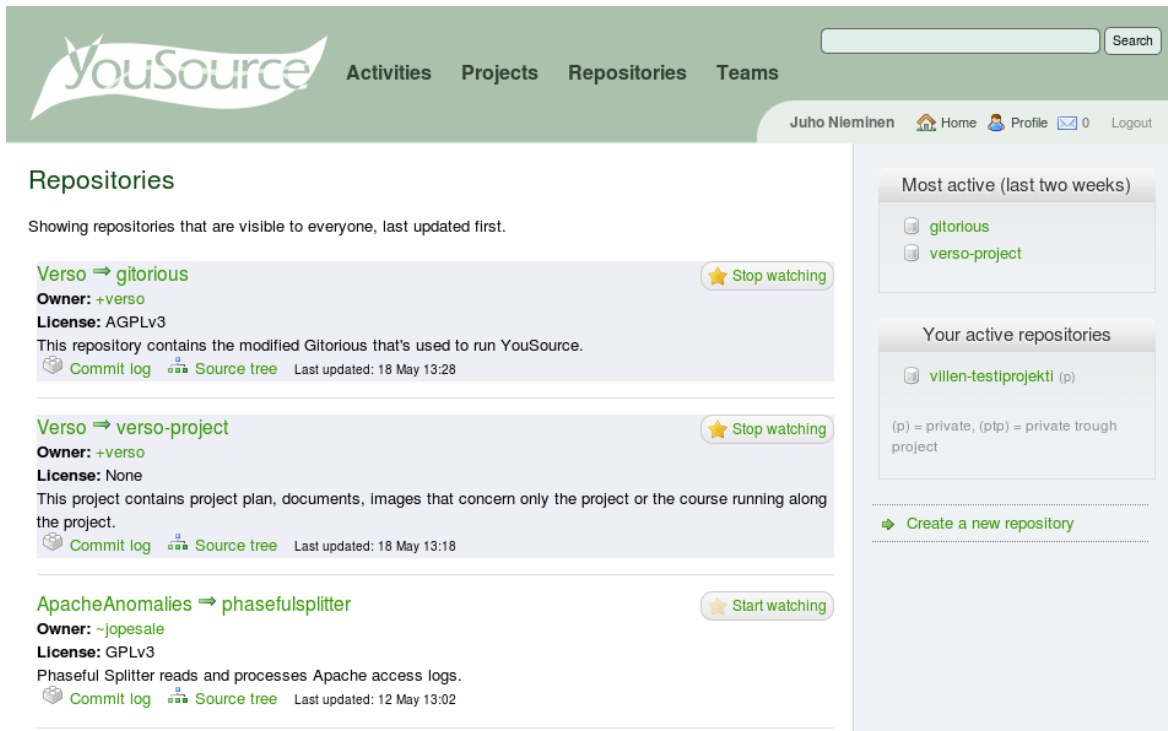
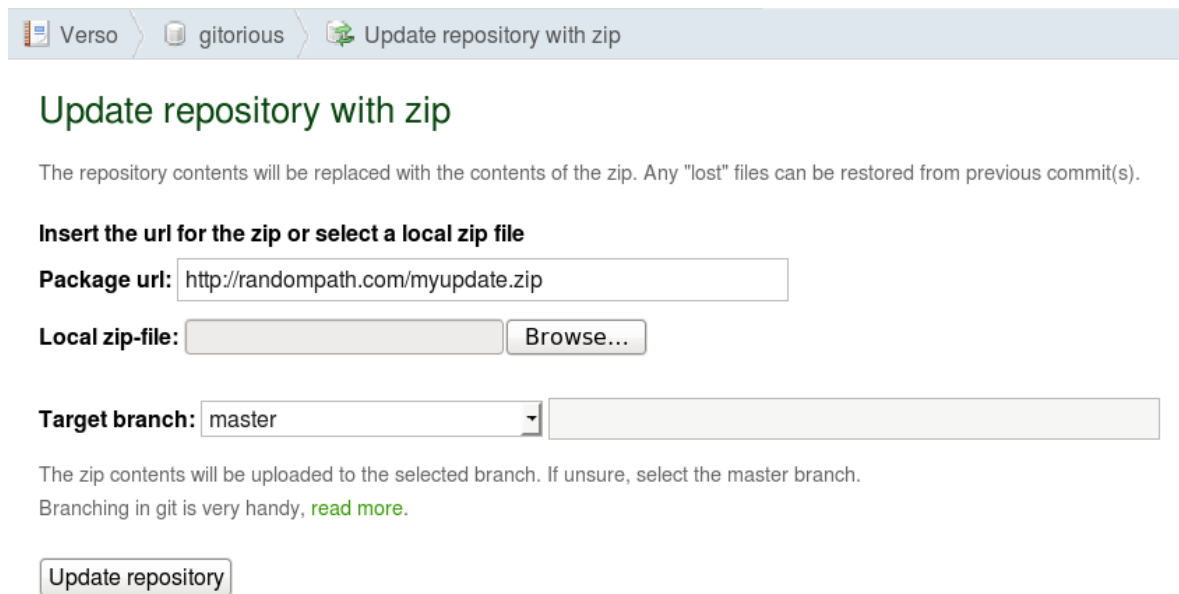


Figure 4.5: The repository browser (*repositories* page).

## 4.7 Updating Repository with a Zip Package

On the repository page the sidebar contains links to operational pages concerning the repository. One of these links is *update repository with zip* which leads to a form on a new page (see Figure 4.6). The form accepts a URL of a zip file or a path to a local zip file for the update process. The form is also asking the user to specify the branch in which does he want 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.



The screenshot shows a web interface for updating a repository. At the top, there is a breadcrumb trail: 'Verso' > 'gitorious' > 'Update repository with zip'. Below this is the title 'Update repository with zip' in green. A paragraph explains that repository contents will be replaced by the zip file, and any 'lost' files can be restored from previous commits. The form has three main sections: 1. 'Insert the url for the zip or select a local zip file' with a 'Package url:' field containing 'http://randompath.com/myupdate.zip' and a 'Local zip-file:' field with a 'Browse...' button. 2. 'Target branch:' with a dropdown menu set to 'master' and an empty text input field. 3. A paragraph explaining that the zip contents will be uploaded to the selected branch, with a link to 'read more' about git branching. At the bottom is an 'Update repository' button.

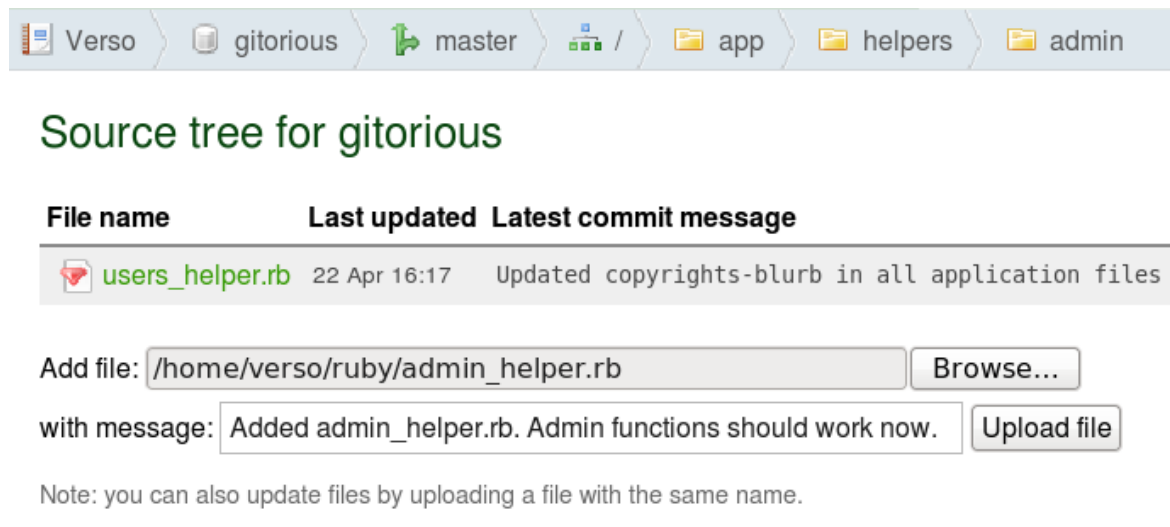
Figure 4.6: The view of updating a repository with a zip file.

## 4.8 Adding Single Files to a Repository


Gitorious offers a way to browse the contents of a repository with a source tree view. This view was updated so that at the bottom of the page there is a small form which accepts a local file (see Figure 4.7). After sending the form the file will be added to the directory that the source view is displaying. The current directory is shown in the breadcrumb. The branch where the file will be added is the branch where the repository head is currently in. This form also allows updating files if a file with existing file name in the repository is given.

## 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) or later on the repository edit page. A private repository means that it will only be shown to people that have view rights to it. A viewer is a new type of user privilege that was developed into YouSource. The owner of a repository can modify who can view his repository in the *manage collaborators* page. The view rights can be given to a user or a team (latter will set all the members of the team as viewers). However, a viewer can only access the repository page, not the repository data. Those privileges require committer or administrator rights to



The screenshot shows the source tree for the 'gitorious' repository. At the top, there is a breadcrumb navigation path: Verso > gitorious > master > / > app > helpers > admin. Below this, the title 'Source tree for gitorious' is displayed. A table lists the files in the current directory:

File name	Last updated	Latest commit message
 users_helper.rb	22 Apr 16:17	Updated copyrights-blurb in all application files

Below the table, there is a form for adding a new file. It includes a text input field for the file path, a 'Browse...' button, a text input field for the commit message, and an 'Upload file' button. The current values are:

Add file:

with message:

Note: you can also update files by uploading a file with the same name.

Figure 4.7: The form for adding single files to the current branch.

the repository.

## 4.10 Private Project

Much like repositories, projects can be marked as private as well. This can be done when the project is created or later on the project edit page. 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 4.8).

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 which is only visible to members is also visible to users who have gained view rights to one of the repositories in the project. That kind of user still can't view the other repositories in the project.

## 4.11 Editing of Repository Metfiles

Not all user interface changes concern the WWW interface. One improvement was implemented to the use of command line interface with YouSource. The repository name, description and the option to allow merge requests are editable through



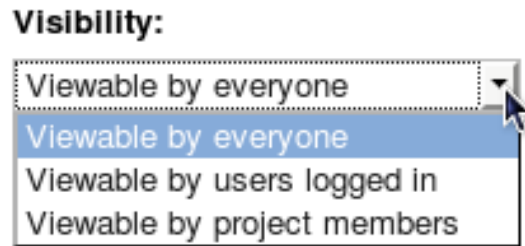


Figure 4.8: A partial view of the create a new project page showing the options for creating a private project.

the command line interface. This is achieved by adding a branch called `yoursource_metafiles` to every repository created in YouSource. The branch contains a plain text file which specifies the description and a YML file which sets the other options. These files are synchronized with the repository's options so that changes in the mentioned files show up in the repository's info 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 people to use the application. This is why the customer in Verso project wanted to develop the user interface of the new software to be more user friendly than the prototype. Partly, this is evident 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. The Verso group found problems on their own by using the application during the development. The 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 were made to the header and footer from which the header was totally redesigned. The menu in header was centered and the user control links were separated from the main menu. Auri Kaihlavirta supplied a logo, 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 4.9.

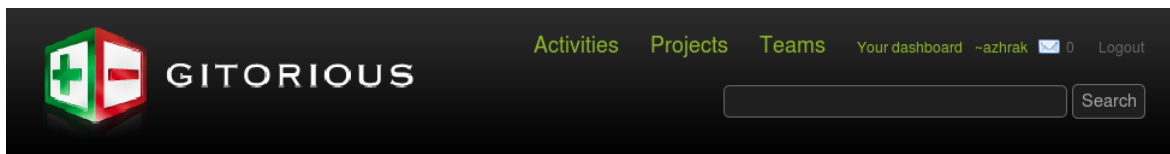


Figure 4.9: Site header of Gitorious when logged in.

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 a 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.

The Verso group found a few usability flaws in Gitorious during their own use of the software. Buttons to create a new project and a new repository were added to project and repository pages respectively. Buttons to delete a project or 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 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, the *projects* and *repositories* pages got a clarifying hint stating the order of the items listed on the pages. The directory table in the *source tree* page was supplemented with headings. In the *repository* page sidebar the link *repository clones* was made to only appear as a

link 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 number of external Ruby libraries (most notably Grit to handle Git functions).

### 5.1 Inner and External Interfaces

Let us we examine the interfaces used in YouSource from the point of view of the source code. 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.

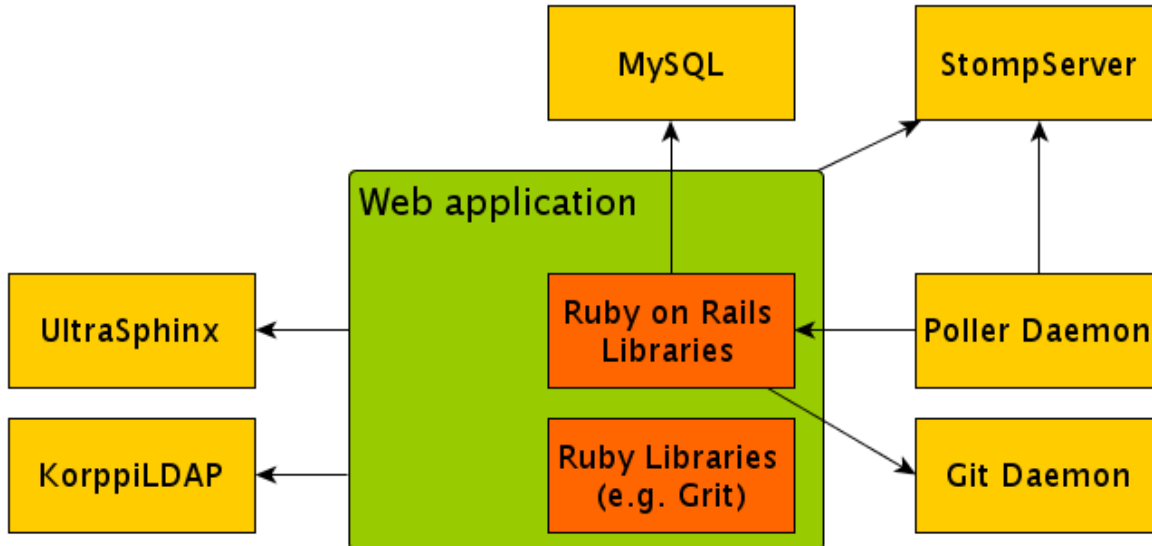


Figure 5.1: The components used in YouSource and their relations to each other.

Examining the interfaces 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. Everything else does not show for the user at all which means they are inner interfaces from the user's point of view.

## 5.2 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 serves that service if it is enabled. Git daemon makes it possible for users of YouSource to push into repositories and clone them with a right URL.

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

**KorppiLDAP** is an authentication and directory access method. 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 user information (except passwords), event information, project information and repository information.

**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 on Rails libraries** provide the basic functionalities for 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 listed above in the description of Poller daemon.

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

### 5.3 Class Structure

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`, `controller`) and data display classes (`view`). Besides models, controllers and views, there are `processor` classes that handle queued tasks given by controllers and `helper` classes that help views to process display data. Other types of classes are also used from external libraries.

The class structure of YouSource is presented in Figure 5.2. It shows the `Repository` model and the classes that are the most related to it. Other models in YouSource have similar structures. In the figure, the classes marked red, `ActiveRecord.rb` and `ApplicationController.rb`, indicate Ruby on Rails base classes. The classes marked green, such as `Repository.rb` and `RepositoriesController.rb`, indicate classes that were modified during the project.

Not all models were modified as extensively as the `repository` model. In the MVC architecture, most of the modifications were done to the views as usability improvements. Figures 8.1 and 8.2 in Section 8.1 show the modifications done to Gitorious in Verso project.

### 5.4 Metafiles

Verso project developed to YouSource two metafiles which describe information of a repository. A YAML file holds information about the repository name and the possibility to allow merge requests. Another plain text file holds the repository description. The YAML format of file `repository.yml` is as follows:

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

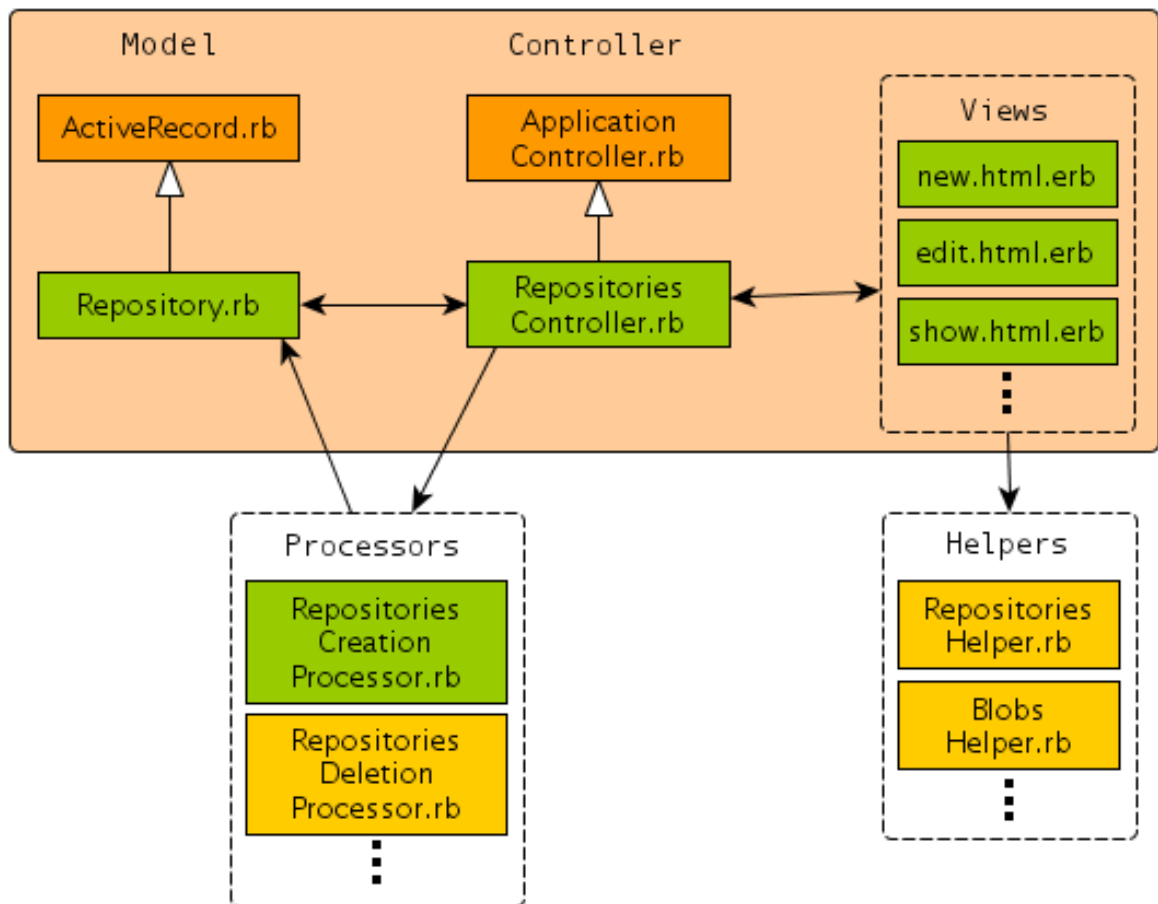


Figure 5.2: The class structure of YouSource from the point of view of `Repository` class.

## 6 Programming Practices

The programming in Verso project started in the first phase of the project after Gitorious was chosen as a platform for the development. From the start, 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 concern when formatting the source code. The Verso group followed their own formatting practices which were influenced by the original Gitorious source code. Two whitespaces were used as indentation between code blocks. Only one command was written per line. Long lines were split to several lines using indentation to emphasize the continuity of the command. When possible, conditional commands were written using positive logic utilizing the possibilities of Ruby such as the `unless` statement. SQL statements were made simpler by using the `find` function of `ActiveRecord` class of Ruby on Rails.

All the names in the code were written in English. The naming of methods and variables was kept as self-explanatory as possible. The idea was to keep the naming good so no clarifying comments would be needed. Multipart names were written with underline characters between the parts. Class names were an exception and multipart class names were written together with every part starting with a capital letter. Constants were written with all capital letters.

Commenting was kept minimal. The aim was to keep the code tidy and readable so that no comments would be needed. However, when something peculiar was done, it was accompanied with an explaining comment. Another situation when commenting was used is when the implemented solution is considered imperfect. These parts were marked with `FIXME` comments to help the future developers find the problem areas. Comments were written in English.



## 6.2 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.

## 6.3 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 most 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.

## 6.4 Source Code Example

Below is an example of source code formatting 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
end
```

## 7 Testing

The developed YouSource software, was tested during the project by the Verso group, customers, 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 most of the key features were implemented. The usability tests took place at the final phases of development.

The 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 the 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.

### 7.1 Integration Testing

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 seeing how it worked. After these first integration tests were made, a functional feature was then implemented on the test server which was accessible for the group of initial users and the customers. Then then 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 new features were implemented and tested one at a time.

The tests were based on the functional requirements specified by Verso group. The needs and goals for the requirements were specified by the customer. 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

YouSource's ease of use was tested in two usability testing sessions reported in [9] and [8]. 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 consisted of two testers from the Verso group, a user and 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 that his mission was to publish something on the YouSource website. 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 publishing was conducted via Git, and for that the user was instructed to use Putty to connect to a server called `charra.it.jyu.fi` that had Git installed.

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

### 7.3 System Testing

The functionalities of the application were tested in two system testing sessions. System Test Plan [3] describes the tests that were carried out in the sessions and System Testing Reports [5] and [5] discusses the results from the testing. The system tests go through all the features that were developed during Verso project. Every test has a set preconditions, test steps and postconditions that all must pass for the test to pass. The results display a list of the performed tests, a mark if the test passed or failed and a comment if the test failed.

### 7.4 Testing Results

Unit and integration testing of new features was very useful in finding errors. For example, the visibility of projects and repositories was tested a long time because the more it was tested, 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, like when a file was begin added to a former version of the repository. Turned out that Git made a new branch for the file starting from the former version.

The daily use and testing of the application by the Verso group revealed several issues. Some of them were discovered by accident and some came up as improvement ideas for already implemented features. One example of a feature that got redesigned 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 project and repository creation. Section 8.4 explains this issue more.

An example of a problem that was discovered by accident during the daily use was the deletion of user or team avatar. That feature was a part of Gitorious and it was not a required feature. However, it was discovered, that the deletion of avatar images didn't work properly. The problem was solved and 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.

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 strated 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 for confirmation from users that are added to a team, and a page where a user can get information about how to get started with the application.

In the system testing session two errors were discovered. One concerned private projects and their visibility. If a project was marked to be visible only to users that are logged in, it still showed the project in search results even if the current user wasn't logged in. The other error was about 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 left open for future development.

The instructor Jukka-Pekka Santanen tested the WWW interface of YouSource and discovered several usability issues. Some of these are described in Section 4.12.

Santanen's testing brought up many good improvements and not all of them could be implemented because of lack of time, but the rest were marked down for future development. In addition, one bug was found as a result of Santanen's 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. This fix was also offered to Gitorious and it was approved.



## 8 Realization of Objectives

This chapter describes the requirements set to YouSource and how well they were achieved. The main features that were developed were: login using KorppiLDAP, private projects and repositories, a repository browser page and different updating and mirroring methods for repositories. Some of the features that were left for future development were: possibility to comment and certificate source code, button for automatic merging, editing of text based files via the WWW interface and requiring the user's SSH key only when truly needed.

### 8.1 Realization of Requirements

All (11 out of 11) of the mandatory and nearly all (4 out of 6) of the important functional requirements [2] of the application 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 left out because of lack of time. Out of the features marked as useful only the abbreviation of the repository's license was implemented.

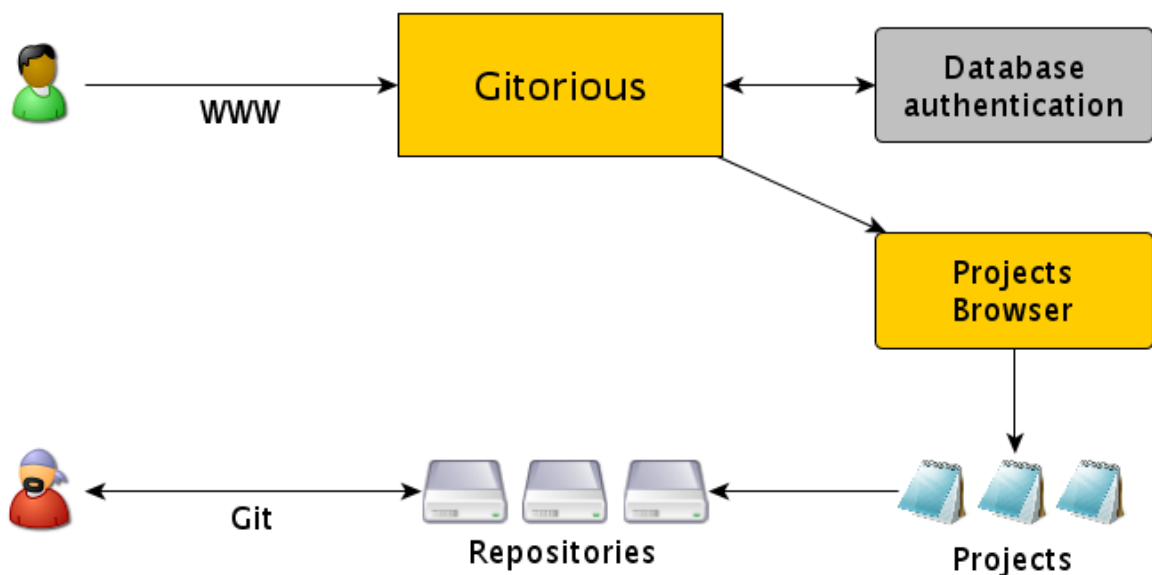


Figure 8.1: Stripped structure of Gitorious at the start of Verso project.

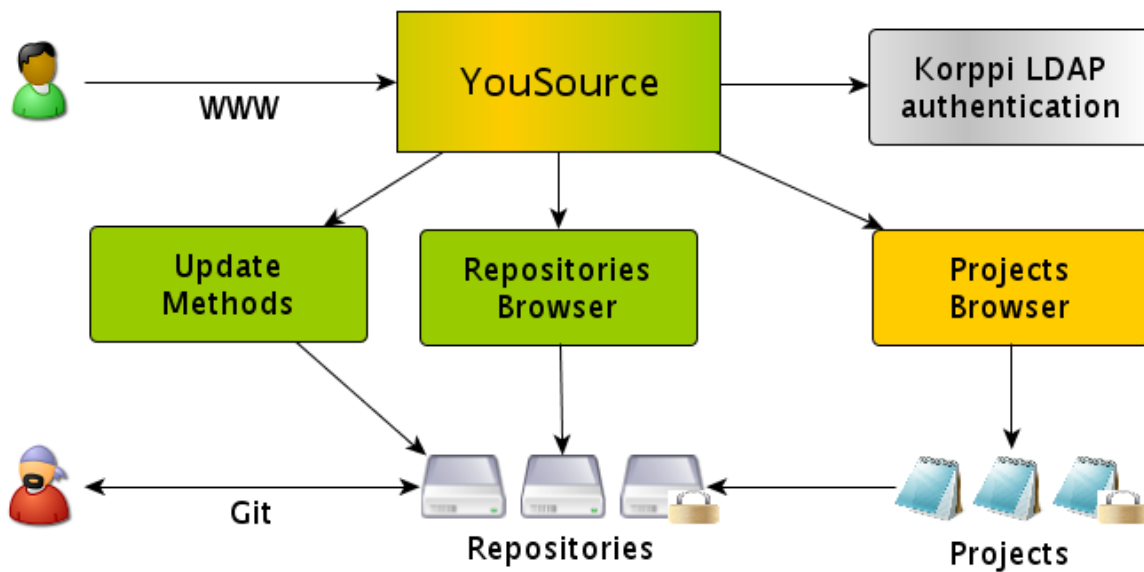


Figure 8.2: Stripped structure of YouSource showing the modifications to Gitorious at the end of Verso project.

## 8.2 Unsatisfactory Solutions in Implementations

The SVN mirroring for a repository was a requested feature from the customer. The idea was to give a way to share source code on YouSource also for the people 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 if they respond. The validation should be expanded so that it checks if the provided URL really leads to a zip package or a SVN repository. In case of 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 for the metafiles. It is used to update the metafile information to the the database. However, the technical instructor Antti-Juhani Kaijanaho pointed out that the metafile helper (`GmrfHelper`) is not actually a helper class in the sense of Ruby on Rails because helpers are normally classes that aid views to display data to the user. The metafile helper should be implemented to the repository module.

Yet another poor implementation is the *repositories* page, also known as the repository browser. 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 the Verso group didn't have enough knowledge of the routing in Ruby on Rails so that the repositories browser could have been routed through the repositories controller. Gitorious developers had made the repositories routing in an odd way and the Verso group didn't know how to modify the repository routes needed for the repository browser.

### 8.3 Complications 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. In addition, KorppiLDAP logging feature didn't work at all on the work stations but on the test server it was fine. Therefore, some bugs were not spotted but until they were out for the users to find. Luckily, Gitorious served a feature that let all the errors happening on the test server to be emailed to the Verso group. This feature enabled a quick response to errors and helped to discover errors that were hard to find.

Gitorious has a problem when the application is restarted. There is a chance that Poller daemon won't shut down in the restating process but instead another Poller daemon is created which results in two Pollers running at the same time. Until this issue was found, there was some hours lost because, for example, creating repositories didn't work properly if more than one Poller was running.

New working tool Git, a new programming language Ruby and a new programming framework Ruby on Rails were also a challenge in Verso project. None of the Verso group had experience of any of these before the project. However, all these turned out to be very manageable tools and learning them was not a big issue. It might be that partly because of these tools, the Verso group was able to achieve a good work flow that might not have been possible with more inflexible tools.

## 8.4 Modifications in Implementation during the Project

Creating a project and a repository on a single form was changed to be handled in two separate forms after it was discovered that error handling was very difficult to implement to a single form 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 together. Instead, a function was implemented that, if a user with no projects comes to the *create a new repository* page, it offers a link to the *create a new project* and a button to create a generic project for the user 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.

## 8.5 Further Development Ideas

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

**Enable user authentication with Kerberos.** This would enable users from other universities to log in to YouSource and collaborative projects to expand more easily.

**Require SSH 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.

**Editing of text based files 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. This feature could be expanded to all text based files in all repositories. Furthermore, for source code files a javascript based programming API (like Bepin [7]) could be helpful

**Certification of source code** Certain users would be given a status that lets them

certificate source code files or commits. This would be a sign to other users that this piece of code is good and it does what it is supposed to.

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

**Possibility to release static files** - This means that YouSource would offer the users to publish files that have no version history. This would be useful if one wants to release a single version of his ...

## 9 Guide for Future Developers

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

### 9.1 Important Bugs

YouSource currently has a few bugs and some of them limit the usability of the software disturbingly. These bugs should be high priority when further development is started.

- SVN mirroring is not working on the test server.
- When a team member is removed, he can still create a repository under the team's project.
- *Activities* page crashes if an active project is removed.

### 9.2 Improvements to Existing Features

A good place to start improving the existing features is to go through the unsatisfactory solutions in implementations in Section 8.2. More improvements are described below.

- Implement a possibility to add a commit message when updating a repository with a zip package.
- Make the adding of a SSH key before project or repository creation as a part of the creation process.
- Improve the *getting started message* to include more information about git usage such as command `git init` if `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 summits. Those features should take priority when planning for new feature implementations. Below are some other feature ideas that are considered the most useful at this point of the development.

- Implement a how-to-get-started page that describes the most essential features of the application and points the user to the right direction.
- Highlight a missing license of a repository.
- Implement sorting functions to *repositories*, *projects* and *teams* pages.
- Implement a possibility to add keywords to repositories.

### 9.4 Advice for Development

In the Project Report [12] in Section 14.1 "Mitä tekisimme toisin? / What would we do differently?" there are described 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 are

1. **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 helps to keep the version history clean and merge requests to upstream are easier to do.
2. **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 publishing software called YouSource for the Department of Mathematical Information Technology in University of Jyväskylä. YouSource is a web application developed with Ruby on Rails and based on an 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 new features to Gitorious such as private projects and repositories, alternative update methods and user authentication with Korppi credentials.

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.



## 11 References

- [1] GitHub Inc., *Secure source code hosting and collaborative development*, URL: <http://github.com>, 2010.
- [2] Tero Hänninen, Juho Nieminen, Marko Peltola and Heikki Salo, *Requirement Specification in Trac, Verso Project*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.
- [3] Tero Hänninen, *Test Plan, Verso Project*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.
- [4] Tero Hänninen, *System Testing Report, RC-1, Verso Project*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.
- [5] Tero Hänninen, *System Testing Report, RC-2, Verso Project*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.
- [6] Tero Hänninen, *Comparison of Platforms, Verso Project*, University of Jyväskylä, Department of Mathematical Information Technology, 2010
- [7] Mozilla, *Bespin, Mozilla Labs*, URL: <http://mozillalabs.com/bespin/>, 2010.
- [8] Juho Nieminen, *Muistio, 1. käytettävyydestaus, Verso Project*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.
- [9] Juho Nieminen, *Muistio, 2. käytettävyydestaus, Verso Project*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.
- [10] 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, URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.163&rep=rep1&type=pdf>, 2002.
- [11] Heikki Salo, *Projektiraportti, Verso Project*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.
- [12] Heikki Salo, *Projektiraportti, Verso Project*, University of Jyväskylä, Department of Mathematical Information Technology, 2010.

- [13] Johan Sørensen, *About Gitorious*, URL: <<http://gitorious.org/about>>, 2009.