

Verso Project

Application Report

Tero Hänninen
Juho Nieminen
Marko Peltola
Heikki Salo

Version 0.2.3
Public
26.5.2010

University of Jyväskylä
Department of Mathematical Information Technology
Jyväskylä

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

Project Contact Information

Authors:

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	Modifications	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 was 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	Appearance was fixed and notes concerning the contents were added.	JN
0.1.4	6.5.2010	The realization of objectives chapter was started.	JN
0.2.0	18.5.2010	Figures were added to user interface chapter. Background chapter was modified. Sitemap and metafiles subsection 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	Subsection Header and Sidebar were put together in Page Structure chapter. Inner and external interfaces figure was added and contents for its chapter was written. System testing and unsatisfactory implementations subsections were written.	JN
0.2.3	25.5.2010	User interface chapter was modified and expanded. Programming practices chapter was written and realization of objectives chapter was continued.	JN

Contents

1	Introduction	1
2	Terminology	2
2.1	General Terms	2
2.2	Verso Specific Terms	2
3	Background	4
3.1	Publishing Channel for Source Codes	4
3.2	Gitorious as a Starting Point	4
4	User Interface	6
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	11
4.6	Repository Browser	11
4.7	Updating Repository with a Zip Package	12
4.8	Adding Single Files to a Repository	12
4.9	Private Repository	14
4.10	Private Project	14
4.11	Editing of Repository Metafiles	15
4.12	Usability Modifications to the WWW Interface	15
5	Application Structure	18
5.1	Components	18
5.2	Inner and External Interfaces	19
5.3	Metafiles	20
5.4	Class Structure	20
6	Programming Practices	21
6.1	Formatting Practices	21
6.2	Naming Practices	23
6.3	Commenting Practices	23
6.4	Grouping Practices	24
6.5	Development Platform	24

7	Testing	25
7.1	Testing Practices	25
7.2	Usability Testing	26
7.3	System Testing	26
8	Realization of Objectives	27
8.1	Realization of Requirements	27
8.2	Unsatisfactory Implementations	27
8.3	Complications in Implementation	28
8.4	Modifications in Implementation during the Project	28
8.5	Further Development Ideas	28
8.6	Guide for Future Developers	29
9	Summary	30
10	References	31

1 Introduction

In a work community, sharing information increases productivity [1]. 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 was found unsuitable for the users. 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 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. It was named as YouSource by one of the customers, Tero Tuovinen.

This 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 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 of the application were fulfilled and finally in Chapter ?? the installation guide for the software is presented.

Verso project wrote several documents describing the practices of the project that support this document. The description of Verso project, its goals and practices can be found from the project plan [?]. The realization of the goals and practices is described in the project report [3]. Usability testing sessions are described in two memos [7] [6]. System testing is described in System Testing Plan [9] and System Test Results [10]. The comparison between different software platforms in the beginning of the project is described in Comparison of Platforms [4].

2 Terminology

This chapter explains the alien terms that appear in this document. First, general terms are explained, and then the terms that are more specific to Verso project are described.

2.1 General Terms

Branch	in Git is a pointer to a commit. The current branch determines where the user's new commits will go.
Commit	contains file modification data and a log message from the user describing the changes.
Push	uploads the new commits in the local repository to a remote repository. It can be thought as releasing a version of a code.
MVC architecture	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.
Ruby On Rails	is an open source web application framework for Ruby programming language.

2.2 Verso Specific Terms

KorppiLDAP	is an authentication and directory access method. It is used in YouSource for authenticating the users with Korppi credentials.
Owner	is a user or a team. An owner has one or many projects and repositories. He is allowed to commit to his repositories and modify his projects and repositories details.
Private project	means that the project is only seen by its members.

Private repository	means that the repository is only seen by users that have one of the following rights to it: view, review, commit or administrate.
Project	is a user created project 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.
Public	means that the item (a repository or a project) is seen by all users.
Repository	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.
Team	is a group of users. One or more users are team admins and the rest are team members. A team can own projects and repositories.
Verso group	refers to the developers in Verso project: Tero Hänninen, Juho Nieminen, Marko Peltola and Heikki Salo.
Viewer	is a status that a user can have for a repository. It grants the ability to see the repository even if it is marked as private.
YouSource	is the name of the developed application.

3 Background

This chapter goes briefly through the background of the developed software. What led to starting of Verso project and the development of YouSource? What were the initial goals? The 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.

3.1 Publishing Channel for Source Codes

Currently, at the MIT department 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 done might be lost 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 aim was to get as many people to use proper version control as possible. Therefore, a system that can be used in many different ways to support different users was needed. A WWW interface was planned for the people unfamiliar with version control. A command line interface was meant for the more experienced and even features that will adapt to people's current unique working methods should be implemented, such as reading and mirroring a zip-archive at supplied URL.

3.2 Gitorious as a Starting Point

When the key 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 [4] and finally Verso group ended up with Gitorious [5].

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 problems 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, all the information about the projects and repositories is easily obtained. 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. In addition, the logging in was changed to be similar to 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). Most of the look of Gitorious was left untouched but almost all the new features needed new user interface implementation. This chapter describes the user interface changes that were done to Gitorious. Not only the web pages related to the new features were changed but also minor changes were made here and there to improve usability.

4.1 Sitemap

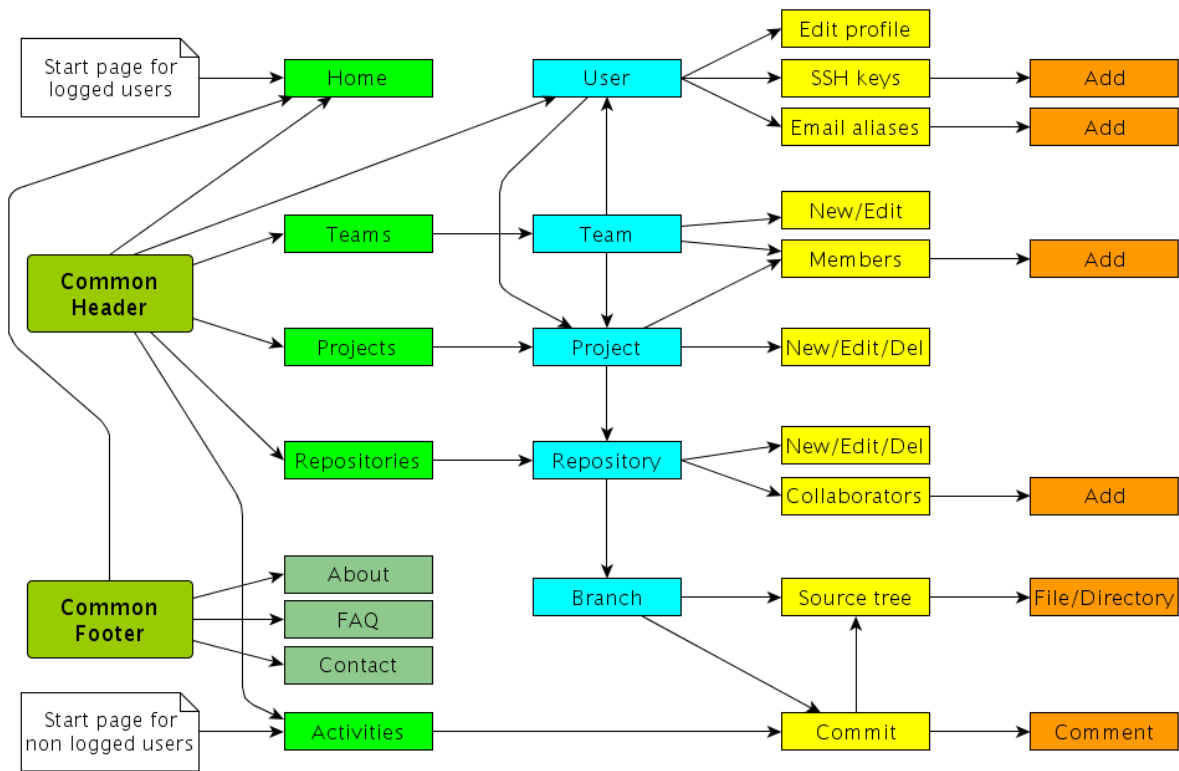


Figure 4.1: Simplified sitemap of YouSource.

Since YouSource is based on a fully functional web application 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 the picture, 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

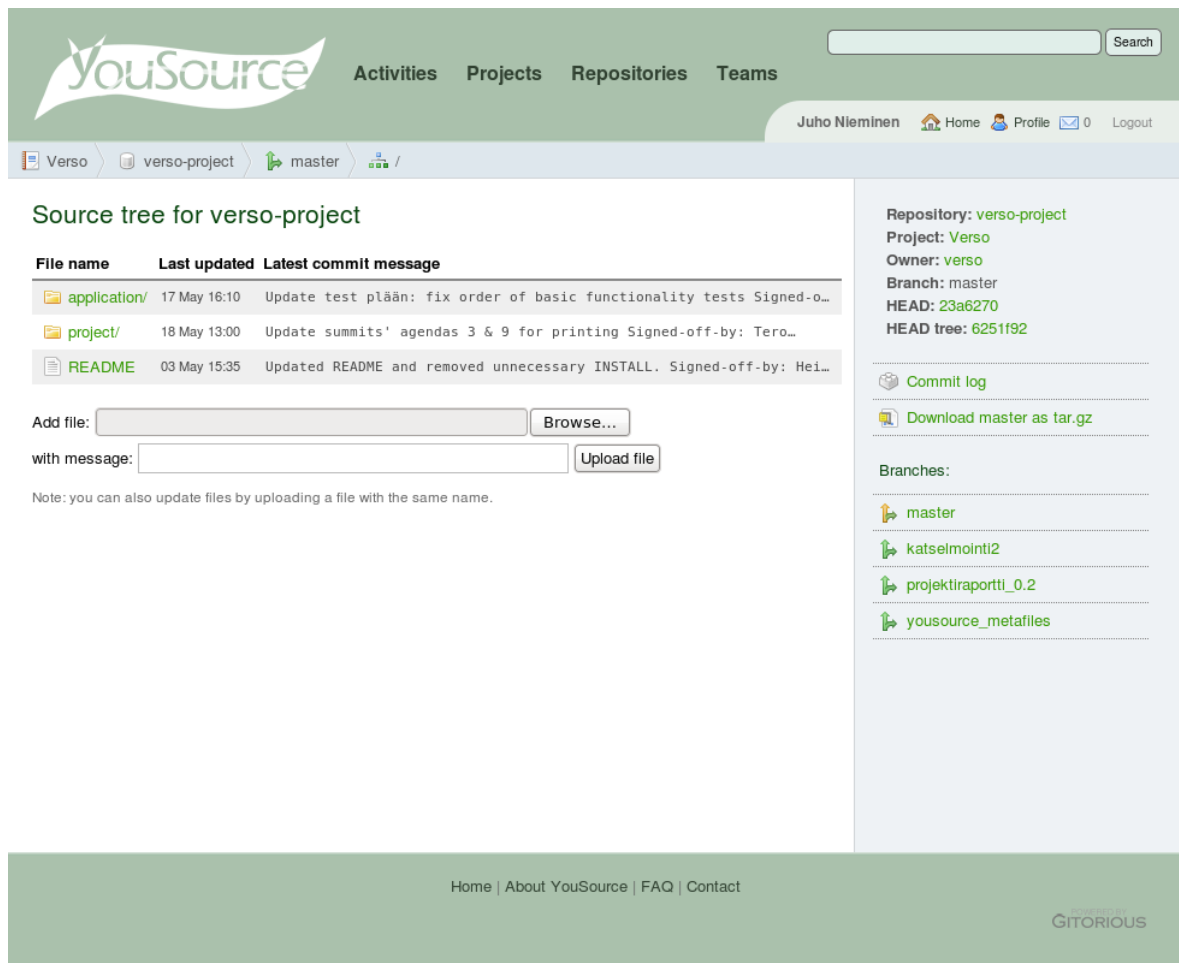


Figure 4.2: The header, breadcrumb, sidebar, main content and the footer on a page.

A general page structure in YouSource consists of five elements: the header, breadcrumb, main content, sidebar and the footer. The header, footer and the breadcrumb sections offer navigation and user control. 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 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.

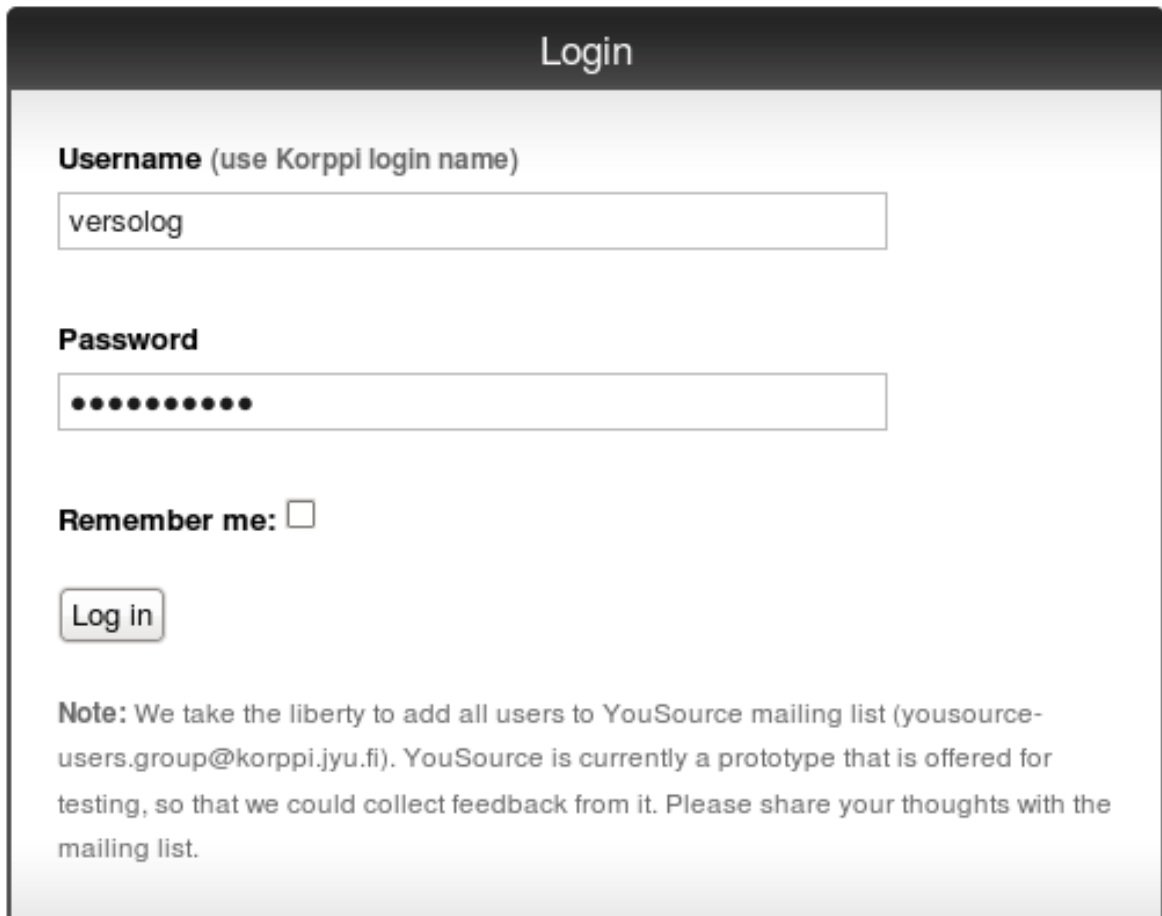
The sidebar is visible on most of the pages in the application. It contains additional information for the main content section and links to management pages.

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.



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 represented by a series of dots.
- Remember me:** A checkbox that is currently unchecked.
- Log in**: A button with the text "Log in".
- Note:** A paragraph of text stating: "Note: 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.

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.

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.

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 highlighted 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.

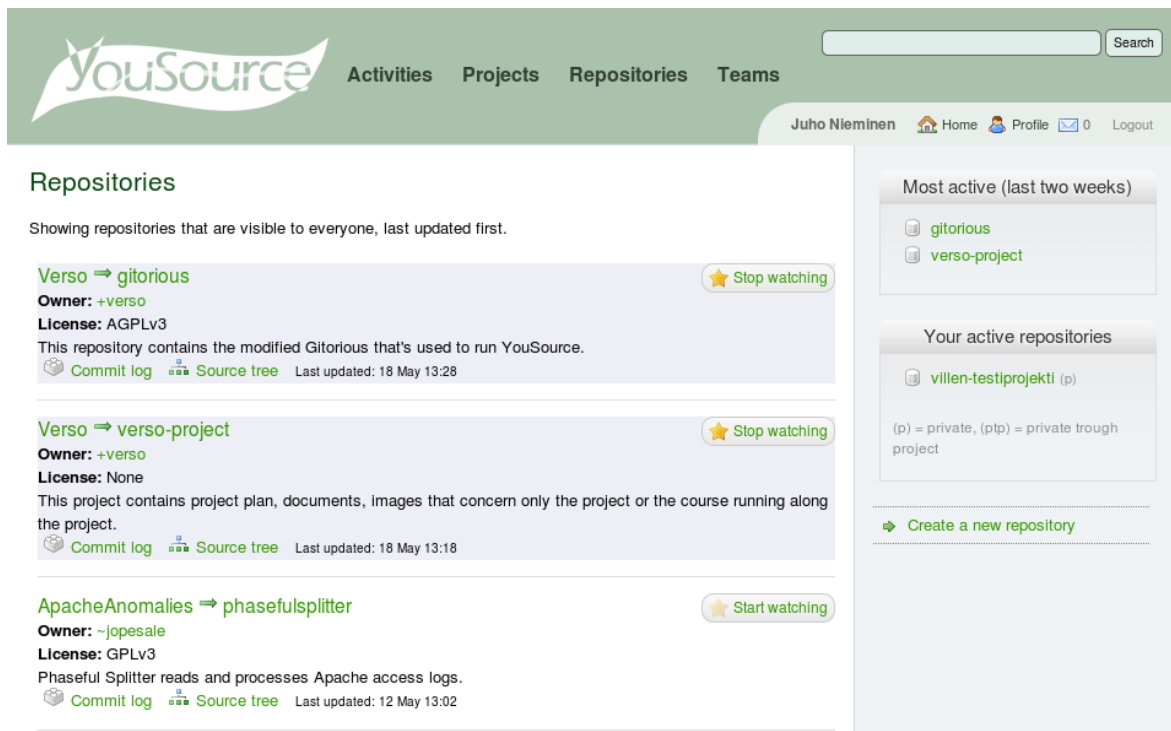


Figure 4.5: The repository browser (*repositories* page).

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.

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

Verso > gitorious > Update repository with zip

Update repository with zip

The repository contents will be replaced with the contents of the zip. Any "lost" files can be restored from previous commit(s).

Insert the url for the zip or select a local zip file

Package url:

Local zip-file:

Target branch:

The zip contents will be uploaded to the selected branch. If unsure, select the master branch.
Branching in git is very handy, [read more](#).

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

repository head is currently in. This form also allows updating files if a file with existing file name in the repository is given.

Verso > gitorious > master > / > app > helpers > admin

Source tree for gitorious

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

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.

4.9 Private Repository

A repository can be marked as private (as opposed to public) when the repository is being created (see Figure ??) 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 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.

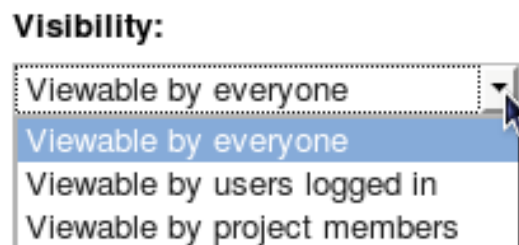


Figure 4.8: A partial view of the create a new project page showing the options for creating a private project.

4.11 Editing of Repository Metafiles

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 the command line interface. This is achieved by adding a branch called `yousource_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. 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.

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 [6]. For this reason, the SSH key help was remarkably improved.

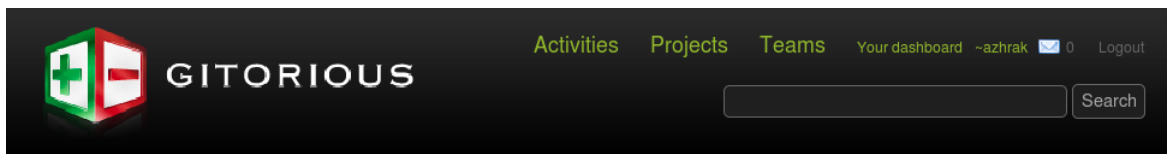


Figure 4.9: Site header of Gitorious when logged in.

The usability testing sessions brought up another issue concerning a help box [7]. 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.

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 machines and it worked fine. It remained unknown why it failed on the test server. The problem was left open for future development.

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. 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

This 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 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 [8], 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.

Poller daemon is a script that is used to execute commands from Stompservers' 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.

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.

- RoR libraries** i.e. 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.

5.2 Inner and External Interfaces

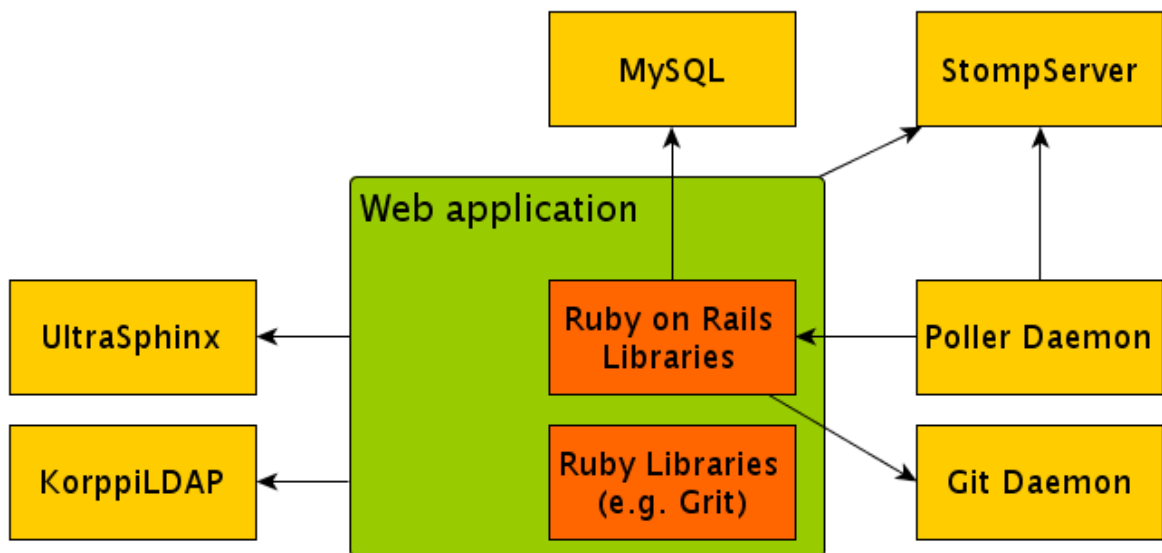


Figure 5.1: The components used in YouSource and their relations to each other.

If 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.

If we examine the interfaces from another angle, 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.

5.3 Metafiles

The metafiles are files that describe information of a repository. In Verso project two metafiles were developed: one YML file that holds information about the repository name and the possibility to allow merge requests, and another plain text file that holds the repository description. The YML file format is as follows:

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

5.4 Class Structure

6 Programming Practices

The programming in Verso project started in the first phase of the project. 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 Practices

Readability was the main concern when formatting the 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.

Below is an example of source code formatting in Verso project taken from `repositories_controller.rb` file.

```
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
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
```

6.2 Naming Practices

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.

6.3 Commenting Practices

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.

6.4 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.

6.5 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.

7 Testing

In this chapter the testing in Verso project is described. The developed software, YouSource, was tested during the project by Verso group, customers, tutors, 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 requirements were implemented. The usability tests took place at the final phases of development.

7.1 Testing Practices

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.

When new features were implemented they were instantly tested by its developer and at least one other project member. The tests were based on the functional requirements supplied by the customer. No automatic testing was exploited, although the Ruby on Rails environment supports it. After the initial testing on the work machines the new features were added to the test server which was accessible for the group of initial users and the customers.

Occasionally the test server turned testing into a challenge. The problem was that it behaved differently compared to the work machines. A fine working feature on a work machine would present an error at the test server. 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 Verso group. This feature enabled a quick response to errors and helped to discover errors that were hard to find.

7.2 Usability Testing

YouSource's ease of use was tested in two usability testing sessions [6][7]. 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 two usability testing sessions were conducted very much the same way but the test users very different. The first session had a test user who was an active programmer and had used to operate with command line and version control. The second testing session had an inexperienced test user instead. The 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.

7.3 System Testing

The overall performance of the application was tested during one system testing session. System Test Plan [9] describes the tests that were carried out in the session and System Test Results [10] 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.

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 of the mandatory and nearly all of the important feature 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 feature that shows an abbreviation of the repository's license was implemented.

8.2 Unsatisfactory 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 machines 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.

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 machined compared to the test server, for example the SVN mirroring. In addition, KorppiLDAP logging feature didn't work at all on the work machines but on the test server it was fine.

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, 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.

8.5 Further Development Ideas

Trac tickets...

8.6 Guide for Future Developers

Trac tickets... Priorities and bugs...

9 Summary

Summing up the whole document...

10 References

- [1] Sanna Talja, *Information sharing in academic communities: Types and levels of collaboration in information seeking and use*, 2002, *New Review of Information Behavior Research*, vol. 3, pages 143–160.
- [2] Tero Hänninen & Juho Nieminen & Marko Peltola & Heikki Salo, *Requirement Specification*, Verso Project, 2010.
- [3] Heikki Salo, *Project Report*, Verso Project, 2010.
- [4] Tero Hänninen, *Comparison of Platforms*, Verso Project, 2010
- [5] Johan Sørensen, *About Gitorious*, 2009.
- [6] Juho Nieminen, *Memo, 2. usability testing*, Verso Project, 2010.
- [7] Juho Nieminen, *Memo, 1. usability testing*, Verso Project, 2010.
- [8] GitHub Inc., *Secure source code hosting and collaborative development*, URL: <http://github.com>, 2010.
- [9] Tero Hänninen, *Test Plan*, Verso Project, 2010.
- [10] Tero Hänninen, *System Test Results*, Verso Project, 2010.