Unplagged Developers Manual

Building the Plagiarism Detection Cockpit

Term paper for the master project I
Mentoring Teacher: Prof. Dr. Debora Weber-Wulff

Department Economics II
HTW Berlin – University of Applied Sciences

Elsa Mahari (s0534556) <Elsa.Mahari@gmx.de>
Tien Nguyen (s0512510) <s0512510@htw-berlin.de>
Dominik Horb (s0534217) <dominik.horb@googlemail.com>
Benjamin Oertel (s0522720) <contact@benjaminoertel.com>
Heiko Stammel (s0534218) <heiko.stammel@googlemail.com>

Contents

Introduction

Even though the big media coverage and interest in plagiarism in Germany has very much subsided after Minister Guttenberg had to resign, because of the plagiarisms found in his doctoral thesis(Google, 2012), the initial idea for the creation of the "Unplagged" project, whose development approach will be described here, can be found in this very case of plagiarism. Related to it were the formation of the GuttenPlag and it's descendent VroniPlag. Both are Wiki-based communities that are collaboratively discovering and collecting plagiarism in their respective cases and are kind of the role models for the way the Unplagged system is developed.

The initial project idea and context were provided by our professor Dr. Debora Weber-Wulff and the two term master project every media informatics student at the HTW-Berlin has to take. As professor Weber-Wulff is a well known expert in Germany on the topic of plagiarism, does research in this field for over ten years(Spiegel-Online, 2011) and is actively involved in the VroniPlag community, she came up with the idea to build a dedicated system — a "Plagiarism Detection Cockpit"(Weber-Wulff, 2011) — that is modeled after the experiences that were made with the workflow used in VroniPlag and GuttenPlag.

So, to put it in a catchy marketing phrase, here is what Unplagged aims to become:

> **Unplagged is a simple, web-based, collaborative system to help discovering, collecting and documenting plagiarism in scientific papers.**

To make things a bit more conceivable, we also often refer to it as a mixture of a very specialised text editor, with a focus on comparing texts and marking passages and a

modern project management tool like Redmine or JIRA, to manage the collaborative aspects of the system. The big distinction we make to other plagiarism software on the market is, that the approach is not to autodetect plagiarism, but focused on aiding the workflow of the users while searching for plagiarized fragments inside a scientific paper, a homework assignment or any other kind of probable textual plagiarism.

This present document will be the handbook that gets you started if you are interested in helping us with the development of this open source project, which is licensed under the GNU GPLv3.

Chapter Overview

One of the biggest problems we faced at the start was, that none of the team members had written a longer scientific text than a bachelors thesis and therefore the experience we got with actual scientific writing was very limited and very specific to the field of computer science. We understand the ethical problems, that come with the betrayal of good scientific practice of plagiators, but we simply can not relate easily to the amount of work that has to be put into a Ph D., or be as passionate about plagiarism as Prof. Weber-Wulff always is, because we never experienced it ourselves.

That is why we had a lot of catching up to do on the most important history behind VroniPlag, the different types of plagiarism, different citation styles and the research Prof. Weber-Wulff and others had already done on systems that try to help finding plagiarism. Chapter 1, A Plagiarism Primer, will give a brief overview of the most important topics to get you up to speed with the domain of the software, if you are not already familiar with it.

Chapter 2 will be the place, where the development process is described and a collection and description of the parts of the system, that already exist or that we identified as necessary parts of Unplagged will be given. As the system is developed with an agile project management style, this will be done primarily based on the user stories.

If you know all those things already and simply want to get started working and coding,

you should probably jump to Developing Unplagged. This chapter will give the technical insights into the system, the basic installation steps and all necessary tools for you as a developer.

Conventions

To markup important words in the text, the following typgraphical conventions are used:

Italic
: First used technical terms

`Constant Width`
: Programm code, file names, paths

`Bold Constant Width`
: Variables that have to be changed by the user

1. A Plagiarism Primer

1.1. Basic Classification of Plagiarisms

1.1.1. Copy&paste

1.1.2. Copy, shake&paste

1.1.3. Patchwriting (rewording)

1.1.4. Structural plagiarism

1.1.5. Translations

1.2. How to detect plagiarism

1.2.1. Software systems

1.2.2. Human approach

1.3. Vroni Plag

2. Project Management and System Requirements

2.1. Project Workflow

To make it possible to work efficiently together in our distributed environment, the team uses Redmine as it's project management tool, which you can access under:

- http://tickets.unplagged.com

If you register there, an administrator should grant you access to the tickets and the wiki, so that you can participate in solving the problems at hand. Our current workflow

2.2. Target Group

2.3. User roles

2.4. Basic functionalities

2.5. Document Parser

2.6. Detection Modes

2.7. Plugin Architecture

2.8. Use Cases

3. Developing Unplagged

Coming from the Project Management and System Requirements here we have yet another set of requirements, before we can start with the actual description of the way you can help us develop the system. This time it's about what we believe will be helpful or sometimes even necessary prerequisites.

First of all, the programming languages mostly used in Unplagged are PHP and JavaScript, both of which in conjuction with a framework. Teaching programming languages is, as you probably can imagine, well beyond the scope of this document, but we will at least try to cover the most important concepts of the frameworks as they occur.

The used frameworks are jQuery for Javascript and ZEND for PHP respectively. jQuery is kind of the industry standard for unobtrusive scripting with about 50% of the Top 10.000 websites using it according to Built With Trends (2012) and the Zend framework is also well established and brings a lot of features, that are useful to this project.

For most of the other topics, we will give you some (hopefully) helpful resources on the way, if it isn't covered thoroughly by us. But just to let you know, here is a list of the buzzwords, er technologies that will be mentioned:

- Scrum
- HTML5 and CSS3
- Continuous Integration
- Responsive Webdesign

- Progressive Enhancement
- Git, Netbeans, Redmine
- Tesseract, Imagemagick
- Simtext

As said in section **??**, the system is developed, so that it should work on multiple platforms. This makes it sometimes difficult to describe certain installation processes in a way that would work for everybody. As it's often most problematic, to get some Linux software running on Windows, we will mostly concentrate on the way those things are done on this platform and give the instructions for other operating systems as an aside if necessary.

3.1. Development Environment

The following parts will mostly focus on the way you can get a development version of unplagged up and running on your system.

3.1.1. Git

The source code and files of all parts of the Unplagged project are managed through Git, with the repository hosted at Github. Git is a distributed version control system, that exists since 2005 and gained more and more track in recent years, because many developers prefer it over other version control system because of its simplicity. This made it also interesting for us to use it for Unplagged.

However, none of the team memebers had ever used Git before in a bigger context so it was a challenge to get it running on all the systems. But we took it, to explore all the features Git offers. It is much easier to create different branches and merge them again, than it is with other version control systems like Subversion.

Installing Git Bash

First of all let's find out, how to install the Git console application, called Git Bash. Unfortunately all the GUIs we were evaluating didn't work consistently, so we decided to use it from the command line only. A very good instruction on how to install the Git Bash can be found on the website of the github project:

Windows: http://help.github.com/win-set-up-git/

Linux: http://help.github.com/linux-set-up-git/

Mac OS X: http://help.github.com/mac-set-up-git/

Getting the source code of the unplagged project

Now it is time to get the project source code on your machine. As said before, the whole unplagged project is hosted on github, so if you want to be able to contribute source code later on, you first need to create an account on https://github.com. This isn't necessary, if you simply want to look into the source code, which can be accessed via https://github.com/benoertel/unplagged.

If you haven't been granted write access to the above mentioned repository by a project member (which is very likely when you are reading this document for the first time), you will need to do a fork of the Unplagged project right at github, like described in:

- http://help.github.com/fork-a-repo/

After this, the following steps are mostly the same for everybody, with the distinction of the project URIs, which should be the one of your newly created fork.

Open up the Git Bash and switch to the directory where you want the project to be located and clone the repository:

Listing 3.1: Cloning a repository

```
1 cd Sites/unplagged.local
2 git clone
      https://<username>@github.com/benoertel/unplagged.git
```

After this you should have a local copy of all the repository data in the specified directory.

The most important git commands

You are now ready to use Git! Here are some more instructions on the most important commands and how to properly use it. However, if the given instructions in this manual are not enough, feel free to checkout the whole Git manual on:

- http://schacon.github.com/git/user-manual.html

The unplagged project consists of several branches, which are used to develop and store code independently of the other developers. Once a new feature is done, it is merged into the master branch. The master branch usually includes only fully tested and deployable source code.

As a new developer, it is important to create an own branch before doing anything else and switch to it.

Listing 3.2: Creating branches

```
1 git branch mynewfeature
2 git checkout mynewfeature
```

Now anything in the repository can be changed, at any point changes can be versioned in the repository by using the *git commit* command. If new files were created, git add has to be executed as well.

Listing 3.3: Commiting

```
1 git add .
2 git commit -m "A message that describes the changes."
```

When the feature is fully working and approved, it has to be merged back to the master branch, in order to get deployed to the staging environment. To do this, the master branch has to be checked out, updated with git pull and then all changes have to be merged from the new feature into the master branch and the feature branch has to be removed.

Listing 3.4: Merging branches

```
1 git checkout master
2 git pull
3 git merge mynewfeature
4 git branch -d mynewfeature
```

In comparison to Subversion, for example Git has one more step to really write back to the repository. After a commit, a push has to be executed, each push can include multiple commits.

Listing 3.5: Pushing to the server

```
1 git push
```

This is nearly it, the changes to the repository have been pushed to the master branch. The only thing, that probably has to be done now, is to open up a pull request on github, if you developed on your own fork of the project. This means, that you are asking the project members who have access to the "real" Unplagged github account, to integrate your changes into the actual project sources. A nice description of how this process is done can be found at github again:

- http://help.github.com/send-pull-requests/

Handling conflicts in merging process

It is possible, if two developers were working on the same part of file, that a conflict raises during the merge. Such a conflict could look like this:

Listing 3.6: Merge conflict

```
1 CONFLICT (content): Merge conflict in readme.txt
2
3 To https://github.com/benoertel/unplagged.git
4  ! [rejected]        master -> master (non-fast-forward)
5 error: failed to push some refs to 'https://github.com/
      benoertel/unplagged.git'
6 To prevent you from losing history, non-fast-forward updates
      were rejected
7 Merge the remote changes (e.g. 'git pull') before pushing
      again.  See the
8 'Note about fast-forwards' section of 'git push --help' for
      details.
9
10 # Unmerged paths:
11 #   (use "git add/rm <file>..." as appropriate to mark
      resolution)
12 #
13 #    both modified:      readme.txt
14 #
```

To resolve the issues, open the files listed in the error message, in this case *readme.txt* and decide how the correct version should look like, by removing all the "< < < < < < < HEAD" and "> > > > > > > b478801d68267ef479acc5ca54544634c52c545c" parts.

Listing 3.7: Creating branches

```
1 <<<<<<< HEAD
2 The goal of this project is the creation of an easy-to-use,
      web-based
3 system to document and detect plagiarism in scientific papers.
4
5 hello world
```

```
6 =======
7
8 The goal of this project is the creation of an easy-to-use,
      web-based
9 system to document and detect plagiarism in scientific papers.
10
11 >>>>>>> b478801d68267ef479acc5ca54544634c52c545c
12 Just a change for educational purposes.
```

Should look like this after merging:

Listing 3.8: Creating branches

```
1 The goal of this project is the creation of an easy-to-use,
      web-based
2 system to document and detect plagiarism in scientific papers.
3
4 hello world
5
6 Just a change for educational purposes.
```

3.1.2. Netbeans

3.1.3. Local Deployment

This subsection will describe how to configure a virtual host properly. A virtual host is a domain that is mapped to the local web server. It is assumed that Apache, MySQL and PHP are already running on the machine. If not, here are some tutorial to get them all running:

Windows: http://www.apachefriends.org/de/xampp-windows.html#1098

Mac OS:
http://www.djangoapp.com/blog/2011/07/24/installation-of-mysql-server-on-mac-os-x-lion/

http://www.quarkstar.at/index.php/2009/05/18/webserver-aktivieren-und-konfigurieren-in-mac-os-x/

The first step is to add the virtual host to the vhost config:

Listing 3.9: Mac OS X: Creating virtual host

```
1 sudo vi /private/etc/hosts
2 #add the following line:
3 "127.0.0.1 unplagged.local"
4
5 sudo vi /private/etc/apache2/extra/httpd-vhosts.conf
```

Listing 3.10: Windows: Creating a virtual host

```
1 open C:\WINDOWS\system32\drivers\etc\hosts
2 #add the following line:
3 "127.0.0.1 unplagged.local"
4
5 open C:\xampp\apache\conf\httpd.conf
6 #Uncomment the following line
7 #Include conf/extra/httpd-vhosts.conf
8 open C:\xampp\apache\conf\extra\httpd-vhosts.conf
```

Add the following configuration to the httpd-vhosts.conf file:

Listing 3.11: Apache configuration

```
1 <VirtualHost *:80>
2 ServerName unplagged.local
3 DocumentRoot "/Users/me/Sites/unplagged.local/public"
4 SetEnv APPLICATION_ENV "development"
5 <Directory /Users/benjamin/Sites/unplagged.local/public>
6 Options +Indexes +FollowSymLinks +ExecCGI
7 DirectoryIndex /index.php
8 AllowOverride All
9 Order allow,deny
10 Allow from all
11 </Directory>
12 </VirtualHost>
```

3.1.4. Tesseract

3.1.5. Simtext

3.1.6. Imagemagick

3.1.7. Continuous Integration

To always have a running version of the latest code, we use an automated workflow, that always deploys everything that has been pushed to the Github repository on the Unplagged staging server. The machine this is done with, is a simple Ubuntu web server, that is also used for hosting our collaboration tools and the webpage.

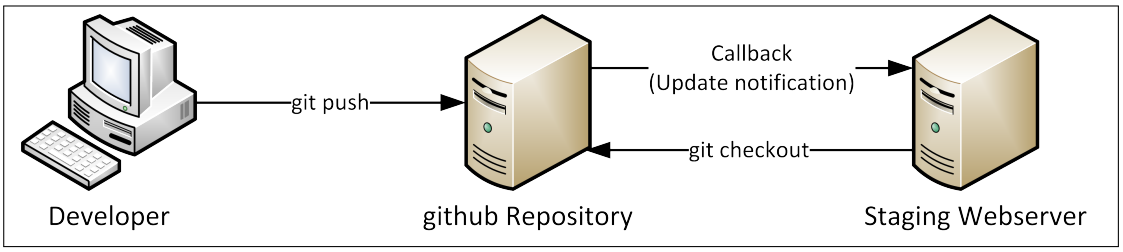

Figure 3.1.: Deployment workflow

As you can see in figure 3.1 the mechanism used for this is a callback, the *post-receive hook* of git, which github employs to let it's users enter a *post-receive URL* to call a URL after someone has pushed to the repository. The URL that gets called is located on our staging server and gets answered by a Redmine plugin called "redmine_github_hook", that would normally only call a checkout on the server-side repository, so that the newest sources can be seen via the Redmine web frontend.

Listing 3.12: Changes to redmine_github_hook.rb

```
1 # Fetches updates from the remote repository
2 def update_repository(repository)
3   command = git_command('fetch origin', repository)
4   if exec(command)
5     command = git_command("fetch origin '+refs/heads/*:refs/
          heads/*'", repository)
```

```
6       exec(command)
7
8       #custom checkout to preview area
9       system('sh /usr/local/etc/scripts/buildUnplaggedPreview.sh
            ')
10    end
11  end
```

However, we tweaked the source code of this plugin slightly, as you can see on line 9 in listing 3.12 so that it also calls the below bash script(listing **??**) to initiate the deployment process.

Listing 3.13: Deployment script

```
1  #!/bin/bash
2
3  cd /var/git/unplagged.git/
4  GIT_WORK_TREE=/var/www/preview.unplagged.com git checkout -f
5
6  cd /var/www/preview.unplagged.com
7  #generate phpdoc
8  apigen -s application/ -s library/Unplagged/ -d docs/phpdoc --
       title "Unplagged Documentation" --todo yes
9
10 #run database build scripts
11 cd scripts/build
12 php initdirectories.php
13 php doctrine_staging.php
14
15 cd /var/www
16 chown www-data:www-data preview.unplagged.com
```

The bash script is then used to do a "clean checkout"(without hidden `.git` folders) of the repository and to run "Apigen", an engine to process the PHPDoc comments inside the project. Those two things can be accessed by the already prepared vHosts on the server:

- http://preview.unplagged.com/
- http://phpdoc.unplagged.com/

If you would like to get access to the preview areas, you need to obtain the password and username from a team member.

Possible Improvements

The above described workflow is, as we believe, already on a good way, but it still has a lot of room for improvement. First of all, it would be nice to only let the deployment go through, if the unit tests ran successfully on the server and to have some sort of email notification mechanism if this wasn't the case.

Another improvement would also be to have a separation into a staging environment with the newest commits and an actual preview environment, that can be deployed to a known stable state/commit of the system in a simple manner.

3.2. Architectural Goals

3.2.1. Progressive Enhancement

3.2.2. Test Driven Development

3.2.3. Responsive Design

A. Sprints

B. Minutes

C. Time Logging

D. Selected Sources

Bibliography

[Built With Trends 2012] Built With Trends, dummy: *jQuery Usage Trends*. http://trends.builtwith.com/javascript/jQuery, 2012. – [accessed online 01.03.12]

[Google 2012] Google: *News Search Interest: plagiat*. http://www.google.com/insights/search/#q=Plagiat&geo=DE&date=1%2F2011%2013m&gprop=news&cmpt=q, 2012. – [Online; accessed 07-February-2012]

[Marcotte 2011] Marcotte, Ethan: *Responsive Web Design*. A Book Apart, 2011. – 150 S.

[Spiegel-Online 2011] Spiegel-Online: *Streit über VroniPlag-Gründer "Verprellter Liebhaber oder SPD-Mitglied, das ist egal"*. http://www.spiegel.de/unispiegel/wunderbar/0,1518,778626,00.html, 2011. – [Online; accessed 26.02.12]

[Weber-Wulff 2011] Weber-Wulff, Dr. D.: *Master Project: Plagiarism Detection Cockpit*. http://www.f4.htw-berlin.de/~weberwu/classes/HTW/projects/Plagiarism-Detection-Cockpit.shtml, 2011. – [Online; accessed 26.02.12]