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

1. Introduction **1**

1.1. Chapter Overview 2

2. The current situation – A plagiarism overview **3**

2.1. Basic Classification of Plagiarisms 3

2.1.1. Copy&paste 3

2.1.2. Copy, shake&paste 3

2.1.3. Patchwriting (rewording) 3

2.1.4. Structural plagiarism 3

2.1.5. Translations 3

2.2. How to detect plagiarism 3

2.2.1. Software systems 3

2.2.2. Human approach 3

2.3. Vroni Plag 3

3. System Requirements **4**

3.1. Target Group 4

3.2. User roles 4

3.3. Basic functionalities 4

3.4. Document Parser 4

3.5. Detection Modes 4

3.6. Plugin Architecture 4

3.7. Use Cases 4

4. Developing Unplagged **5**

4.1. Development Environment 6

4.1.1. Git 6

4.1.2. Netbeans 10

4.1.3. Staging and Preview System 10
4.2. Installation 11
4.2.1. Tesseract 11
4.2.2. Simtext 11
4.2.3. Imagemagick 11
4.3. Architectural Goals 11
4.3.1. Progressive Enhancement 11
4.3.2. Test Driven Development 11
4.3.3. Responsive Design 11

A. Sprints **12**

B. Minutes **13**

C. Time Logging **14**

D. Selected Sources **15**

1. 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.

1.1. 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 2, The current situation – A plagiarism overview, 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.

Although we are using agile methods for the development process, the chapter System Requirements will give a more classical collection and description of the parts of the system, that already exist or that we identified as necessary parts of Unplagged and how we understood the requirements of the VroniPlag workflow.

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 insights into the project workflow, the basic installation steps and all necessary tools for you as a developer.

2. The current situation – A plagiarism overview

2.1. Basic Classification of Plagiarisms

2.1.1. Copy&paste

2.1.2. Copy, shake&paste

2.1.3. Patchwriting (rewording)

2.1.4. Structural plagiarism

2.1.5. Translations

2.2. How to detect plagiarism

2.2.1. Software systems

2.2.2. Human approach

2.3. Vroni Plag

3. System Requirements

3.1. Target Group

3.2. User roles

3.3. Basic functionalities

3.4. Document Parser

3.5. Detection Modes

3.6. Plugin Architecture

3.7. Use Cases

4. Developing Unplagged

Coming from the System Requirements here we have yet another set of requirements for you, before we can start with the actual description of the technologies used for development in the system. This time it's about what we believe will be helpful or sometimes even necessary for the development of Unplagged.

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 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:

- CSS3
- HTML5
- Continuous Integration
- Responsive Webdesign
- Progressive Enhancement
- Git
- Netbeans

- LAMP or similar for your operating system

As said in section **??**, the system is developed in a way, 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.

4.1. Development Environment

4.1.1. Git

The version control of all parts of the unplagged project is managed through Git. Since 2005, Git got more and more famous and many developers prefer it over Subversion because of its simplicity. However, nobody of our team ever used Git before so it was a challenge to get it running on all the systems. But we took the challenge to explore all the features Git offers. It is so much easier to create different branches and merge them again, than it is with other version control softwares like Subversion.

If you didn't use git before, you probably should watch this 8 minutes Git introduction video first: http://www.youtube.com/watch?v=RDGzF2M-zlo

Installating the Git Bash

First of all let's get started with an introduction of how to install the Git console application, called Git Bash. Unfortunately all the GUIs we were evaluating didn't work as expected, so we decided to use it from the console only. A very good instruction on how to install the Git Bash can be found on the website of the github project:

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

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

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

Getting the source code of the unplagged project

Now it is time to get the project source code on your machine. The whole unplagged project is hosted on github, so first you need to create an account on https://github.com. And then go to the directory where the project shall be located. An example for Mac OS X:

Listing 4.1: Cloning a repository

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

The most important git commands

You are 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 manuall 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 indepdently 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 into it.

Listing 4.2: Creating branches

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

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

Listing 4.3: Creating branches

```
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 4.4: Creating 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 4.5: Creating branches

```
1 git push
```

This is it, the changes to the repository have been pushed to the master branch.

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:

```
CONFLICT (content): Merge conflict in readme.txt

To https://github.com/benoertel/unplagged.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/benoertel/unplagged.git
To prevent you from losing history, non-fast-forward updates were rejected
```

```
Merge the remote changes (e.g. 'git pull') before pushing again.  See the
'Note about fast-forwards' section of 'git push --help' for details.

\begin{verbatim}
# Unmerged paths:
#   (use "git add/rm <file>..." as appropriate to mark resolution)
#
#    both modified:      readme.txt
#
```

To reslove 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.

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

hello world
=======

The goal of this project is the creation of an easy-to-use, web-based
system to document and detect plagiarism in scientific papers.

>>>>>>> b478801d68267ef479acc5ca54544634c52c545c
Just a change for educational purposes.
```

Should look like this after merging:

Listing 4.6: 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.
```

4.1.2. Netbeans

4.1.3. Staging and Preview System

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-
http://www.quarkstar.at/index.php/2009/05/18/webserver-aktivieren-und-konfigu

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

Listing 4.7: 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 4.8: 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 4.9: Apache configuration

```
<VirtualHost *:80>
ServerName unplagged.local
DocumentRoot "/Users/me/Sites/unplagged.local/public"
SetEnv APPLICATION_ENV "development"
<Directory /Users/benjamin/Sites/unplagged.local/public>
Options +Indexes +FollowSymLinks +ExecCGI
DirectoryIndex /index.php
AllowOverride All
Order allow,deny
Allow from all
</Directory>
</VirtualHost>
```

4.2. Installation

4.2.1. Tesseract

4.2.2. Simtext

4.2.3. Imagemagick

4.3. Architectural Goals

4.3.1. Progressive Enhancement

4.3.2. Test Driven Development

4.3.3. Responsive Design

A. Sprints

B. Minutes

C. Time Logging

D. Selected Sources

Bibliography

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