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 (s0534217) <Elsa.Mahari@gmx.de>
Heiko Stammel (s0534217) <heiko.stammel@googlemail.com>
Benjamin Oertel (s0522720) <benjamin.oertel@me.com>
Dominik Horb (s0534217) <dominik.horb@googlemail.com>
Tien Nguyen (s0534217) <idontwant2missathing@yahoo.com>

Contents

1. Preface

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, 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[**?**] and is actively involved in the VroniPlag community, she came up with the idea to build a dedicated system — a "Plagiarism Detection Cockpit"[**?**], 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 plagiarism.

This present document will be your handbook, if you want to get started helping in 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.

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. The 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 Unplaggedhow 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

4.1. Development Environment

4.1.1. Git

The version control of all parts of the unplagged projects 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:

```
cd Sites/unplagged.local
git clone https://<yourusername>@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 your own branch before doing anything else and switch into it.

```
git branch mynewfeature
git checkout mynewfeature
```

Now you can change whatever you want in the repository.At any point you can store your changes in the repository by using the git commit command. If you created new files, git add has to be executed as well.

```
git add .
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, you have to checkout the master branch, update it with git pull and then merge all the changes from the new feature into the master branch and remove the feature branch.

```
git checkout master
git pull
git merge mynewfeature
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 done, each push can include multiple commits.

```
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:

```
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

Just a change for educational purposes.
```

4.1.2. Netbeans

4.1.3. Staging and Preview System

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