Peer to Peer collaborative editing based on a git repository

BACHELOR THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Software & Information Engineering

by

Stefan Gussner
Registration Number 01527253

elaborated at the
Institute of Information Systems Engineering
Research Group for Industrial Software
to the Faculty of Informatics
at TU Wien

Advisor: Dipl. Ing. Johann Grabner

Vienna, April 7, 2019

Kurzfassung

Über diese Vorlage: Dieses Template dient als Vorlage für die Erstellung einer wissenschaftlichen Arbeit am INSO. Individuelle Erweiterungen, Strukturanpassungen und Layout-Veränderungen können und sollen selbstverständlich nach persönlichem Ermessen und in Rücksprache mit Ihrem Betreuer vorgenommen werden.

Aufbau: In der Kurzfassung werden auf einer 3/4 bis maximal einer Seite die Kernaussagen der Diplomarbeit zusammengefasst. Dabei sollte zunächst die Motivation/der Kontext der vorliegenden Arbeit dargestellt werden, und dann kurz die Frage-/Problemstellung erläutert werden, max. 1 Absatz! Im nächsten Absatz auf die Methode/Verfahrensweise/das konkrete Fallbeispiel eingehen, mit deren Hilfe die Ergebnisse erzielt wurden. Im Zentrum der Kurzfassung stehen die zentralen eigenen Ergebnisse der Arbeit, die den Wert der vorliegenden wissenschaftlichen Arbeit ausmachen. Hier auch, wenn vorhanden, eigene Publikationen erwähnen.

Wichtig: Verständlichkeit! Die Kurzfassung soll für Leser verständlich sein, denen das Gebiet der Arbeit fremd ist. Deshalb Abkürzungen immer zuerst ausschreiben, in Klammer dazu die Erklärung: z.B: „Im Rahmen der vorliegenden Arbeit werden Non Governmental-Organisationen (NGOs) behandelt, ...“. In LaTeXwird diese bereits automatisch durch verwenden des Befehls `\ac` erreicht. Für Details siehe Paket `glossaries`.

Schlüsselwörter

Abstract

About this template: This template helps writing a scientific document at INSO. Users of this template are welcome to make individual modifications, extensions, and changes to layout and typography in accordance with their advisor.

Writing an abstract: The abstract summarizes the most important information within less than one page. Within the first paragraph, present the motivation and context for your work, followed by the specific aims. In the next paragraph, describe your methodology / approach, and / or the specific case you are working on. The third paragraph describes the results and the contribution of your work.

Comprehensibility: People with different backgrounds who are novel to your area of work should be able to understand the abstract. Therefore, acronyms should only be used after their full definition has given. E.g., "This work relates to non-governmental organizations (NGOs), . . . ".

Keywords

Contents

List of Figures

List of Tables

List of Listings

List of Algorithms

1 Introduction

1.1 Problem Description

When multiple people are working on solving a Problem it is not clear who is implementing which part. Current vcs systems conflict resolutions require manual conflict resolution if multiple people have modified the same file. In order for a group to discuss a problem all the code has to be commited and pulled by everyone first. This introduces friction. In order to solve these and other problems real time collaborative editors have been created. But current implementations of real time collaboration editors / editor plugins are unaware of the underlying version control system and therefore are based on the idea of just sharing files of a host mashine or a single source of truth file on a server instead of basing edit histories on versions of files known to the version control system anyway.

In state of the art collaborative code editors it is not possible for a user to easily see who made specific changes. Usually all the changes are bundled in one commit and the accountability is lost. In order to convert this concurrent model to a commit understood by Git it should be possible to stage changes by author. This approach might require more sophisticated logic than just interpreting changes as strings given that a command could be modified by two users and applying only half the changes as part of a commit could result in invalid syntax or semantic.

1.2 Expected Results

Enable real time collaboration on source code based on a Git[1] project by continuous tracking of code changes synchronizing over peer to peer connection. Lowering overhead of splitting tasks by enabling everyone see what other people are working on. Allowing discussions about source code that has not yet been commited. Changes should be commitable by author. In order to be able to compile sourcecode, changes of other people can be toggled off. Using Git as a base enables opportunistic real time collaboration. In other words if a connection is possible changes will be propagated to other people working on the same branch. If not the changes will be sent when a connection is available. [1] Therefore it should be analog to the benefits of moving to a decentralized vcs.

1.3 Motivation

Current implementations of real time collaboration tools are not designed with version control systems in mind.

Transforming real time collaborative edits into regular Git commits by author will reduce a lot of friction in the adoption for real time collaboration software. Using a peer to peer solution with the ability to deal with disconnect events using information already known to the version control system will drasticly increase the ease of use. Ideally a user won't even have to think about using the extension.

[1] https://git-scm.com/

In diesem Kapitel wird der Forschungsbedarf aufgezeigt. Nach dem Lesen dieses Kapitels sollten folgende Punkte klar dargestellt sein:

- Aktueller Stand der Wissenschaft in Bezug auf die zuvor formulierte Problemstellung und klare Darstellung, was hier unzureichend/offen ist.
- GGf. Darstellung des größeren Forschungsbereichs, in den die Diplomarbeit eingebettet ist.
- Darlegung der Bedeutung des Themas für den Stand oder die Weiterentwicklung eines Bereichs der Informatik (z.B. Datenbanksysteme, Mobile Anwendungen, Java-Programmierung, Rechenzentrumsbetrieb, ...) oder eines Fachbereichs (z.B. Bankwesen, Wertpapierhandel, Gesundheitswesen, Transportwesen, Flugsicherheit ...). Erklärung, was durch die Lösung des Problems z.B. kostengünstiger/schneller/hochwertiger/sicherer/anwendbarer/„schöner“ etc. wird.

1.4 Zielsetzung

Nachdem die Problemstellung und die Motivation zu deren Lösung formuliert wurden, wird in diesem Kapitel das zu erarbeitende Resultat beschrieben.

2 Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen und alle in der Arbeit verwendeten und für das Verständnis relevante Begriffe erläutert. Kapitelnamen spezifizieren, anpassen an die Fragestellung der Arbeit.

Gerade im Bereich der Grundlagen wird viel Literatur zitiert – Details zum Zitieren finden Sie im Kapitel 4. Da keine Diplomarbeit so innovativ ist, dass sie nicht auf vorhandenes Wissen aufbaut und in ein entsprechendes Forschungsumfeld eingebettet ist, kommt an dieser Stelle der Literaturrecherche eine besondere Bedeutung zu. Als Daumenregel gilt, dass der aktuelle Stand der Wissenschaft in der Informatik üblicherweise durch Publikationen v.a. der letzten 2 – 4 Jahre repräsentiert wird.

2.1 Aktueller Stand der Technik

In diesem Kapitel wird ein Überblick über bereits existierende Lösungen für die Problemstellung bzw. verwandte Problemstellungen gegeben. Dabei ist eine Klassifizierung der existierenden Lösungen empfehlenswert. Eine Analyse der Lösungen, nach Kriterien sortiert, sollte insbesondere auch die Defizite der existierenden Lösungen erläutern und damit insbesondere auch eine Begründung liefern, warum diese Lösungen für die Problemstellung der Arbeit nicht herangezogen werden können.

2.1.1 Unterkapitel

Bei der Verwendung von Gliederungsebenen gibt es Folgendes zu beachten:

- Es sollten nicht mehr als 3 Gliederungstiefen nummeriert werden.
- Unterkapitel sind nur dann sinnvoll, wenn es auch mehrere Untergliederungen gibt. Ein Kapitel 2.1.1 sollte somit nur dann verwendet werden, wenn es auch 2.1.2 gibt.
- Oft ist es einfacher und besser verständlich, Aufzählungen als Text zu formulieren und somit weitere Gliederungsstufen zu vermeiden.

2.1.2 Abbildungen

Beschreibungen zu Abbildungen und Tabellen stehen unter dem Bild. Jede Abbildung muss im Fließtext referenziert werden. In LaTeXbesitzen Abbildungen typischerweise Labels, welche zum referenzieren verwendet werden. Zudem plaziert LaTeXdie Abbildungen an geeigneten Stellen, was meistens auch wünschenswert ist. Falls das nicht gewünscht wird, kann es durch Optionen beeinflusst werden.

Abbildung 21 verdeutlicht ...
(siehe Abbildung `\ref{<label>}`)

2.1.3 Tabellen

Jede Tabelle muss im Fließtext referenziertw werden. Für Tabellen gelten die selben Regeln, wie für Abbildungen (siehe dazu Abschnitt 2.1.2).

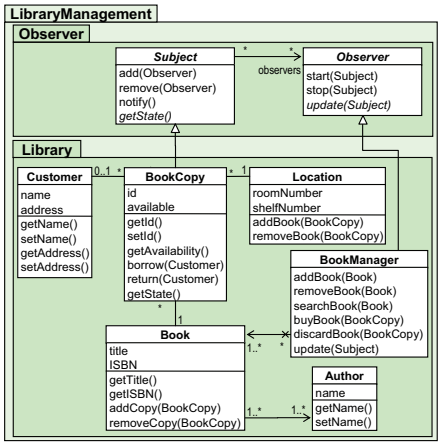

Figure 21: xxx (Quelle zitieren, wenn nicht selbst erstellt)

Linksbündig	Zentriert	Rechtsbündig
Zeile 1	xxx	xxx
Zeile 2	xxx	...
Zeile3	xxx	xxx
	xxx	xxx
xxx		

Table 21: xxx (Quelle angeben)

Eine Beispiel einer Tabelle ist in Tabelle 21 zu finden:

Bitte beachten Sie, dass Tabellen generell so einfach wie möglich gehalten werden sollen. Tabelle 21 dient unter anderem dazu Studierenden zu zeigen, wie Tabellen in LaTeX erstellt werden können und wie Farben verwendet werden.

3 Konkrete Problemstellung – Umfeldbeschreibung

In diesem Kapitel wird die eigentliche Problemlösung in einem oder mehreren Unterkapiteln ausgeführt. Die Strukturierung dieser Kapitel ist naturgemäß sehr stark von der konkreten Aufgabenstellung abhängig. Der Name dieses Kapitels ist anzupassen, z.B. Umfeldbeschreibung – Fallbeispiel . . . , konkreter schreiben je nach Art Diplomarbeit/Fragestellung.

4 Hinweise zur Literatur

4.1 Literatursuche

Der Vollzugang zu einigen Publikationen ist nur intern aus dem TU-Netz möglich. Um auf möglichst viele Papers extern zugreifen zu können, wird von der TU Wien eine VPN-Zugangsmöglichkeit angeboten, diesen VPN-Zugang bitte gleich einrichten.

Besonders ergiebig sind folgende Search-Engines:
Microsoft Academic
ACM-Datenbank
Google Scholar

Wir empfehlen, vor Beginn Ihrer Arbeit einige Diplomarbeiten, die am INSO oder generell an der Fakultät für Informatik verfaßt wurden, zu Ihrem Themenbereich zu suchen und Aufbau, Schreibstil, Art der Abbildungen etc. durchzuschauen. Arbeiten finden Sie hier.

Weitere Datenbanken und Suchmaschinen:
Elektronische Zeitschriftenbibliothek der TU Wien
Scientific Literature Digital Library (CiteSeer)
Ingenta
INSPEC

Journals:
IEEE - Institute of Electrical and Electronics Engineers, Inc. - Library
Verlag Springer - Springer Link
Elsevier

Bibliotheken und Online-Kataloge:
Online-Kataloge des Österreichischen Bibliothekenverbundes
Online-Katalog der TU Wien (ALEPH)
Digital Bibliography & Library Project (DBLP) of University of Trier
The Collection of Computer Science Bibliographies

4.2 BibLatex

Biblatex bietet verschiedene Möglichkeiten an, um Literatur zu referenzieren. Die beiden häufigsten Befehle sind `\cite` und `\citeauthor`.

Beispiele wie referenziert werden kann:
Fankhauser, Schanes, and Brem beschreiben in [2] ...
In [4] zeigen Schanes et al. wie ... Weitere Informationen können in [3] von Oasis entnommen werden.

Wir empfehlen JabRef, um die Literaturdatenbank zu verwalten.

5 Algorithmen und Quellcode

5.1 Beispiele für Quellcode

Beispiel eines Quellcodes ist im Quellcode 5.1 zu finden.

```
1 //Start Program
2 System.out.println("Hello World!");
3 //End Program
```

Listing 5.1: Short code

5.2 Beispiele für Algorithmen

Algorithmus 5.1 dient als Beispiel.

input : A bitmap Im of size $w \times l$
output: A partition of the bitmap

```
1  special treatment of the first line;
2  for i ← 2 to l do
3      special treatment of the first element of line i;
4      for j ← 2 to w do
5          left ← FindCompress(Im[i, j − 1]);
6          up ← FindCompress(Im[i − 1, ]);
7          this ← FindCompress(Im[i, j]);
8          if left compatible with this then ;                    // O(left, this)==1
9
10             if left < this then Union(left,this);
11             ;
12             else Union(this,left);
13             ;
14         end
15         if up compatible with this then ;                      // O(up, this)==1
16
17             if up < this then Union(up,this);
18             ;
               // this is put under up to keep tree as flat as
                  possible
19             else Union(this,up);
               ;                                          // this linked to up
20         end
21     end
22     foreach element e of the line i do FindCompress(p);
23 end
```

Algorithm 5.1: Sample algorithm

6 Ergebnisse

Die Resultate der Arbeit präsentieren und nach Möglichkeit aussagekräftige, eigenständige Abbildungen einbauen. Namen des Kapitels konkretisieren, an jeweilige Arbeit anpassen – Lösungsvorschlag/Implementierung im Titel des Kapitels benennen.

7 Zusammenfassung und Ausblick

Bibliography

References

[1] Jonathan Sillito Brian de Alwis. „Why are software projects moving from centralized to decentralized version control systems?“ In: *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering* (2009), p. 38.

[2] Florian Fankhauser, Christian Schanes, and Christian Brem. „Softwaretechnik - Mit Fallbeispielen aus realen Entwicklungsprojekten“. In: 1st ed. München: Pearson Studium, 2009. Chap. 13, pp. 589–646. URL: http://www.inso.tuwien.ac.at/publications/softwaretechnik/ (visited on 01/16/2017).

[4] Christian Schanes et al. „Security Test Approach for Automated Detection of Vulnerabilities of SIP-based VoIP Softphones“. In: *International Journal On Advances in Security* 4.1 and 2 (Sept. 2011), pp. 95–105. URL: http://www.iariajournals.org/security/tocv4n12.html (visited on 01/16/2017).

Online References

[3] Oasis. *Organization for the advancement of structured information standards*. 2010. URL: http://www.oasis-open.org (visited on 01/16/2017).

A Appendix

Listings, data models, forms, ...