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 23, 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 Fundamentals

2.1 State of the Art

2.1.1 Teletype for Atom

Teletype for Atom[1] is a Project enabling editing files peer to peer. It is based on [5] [6] [2]. With Teletype it is possible to edit files currently opened by the "host". The files are only persisted on the "host" not on all peers.[2] Therefore disconnecting from the network cuts off the editing workflow. It is not possible to access all the files in a project unless they are opened by the host.

2.1.2 Visual Studio Live Share

Visual Studio Live Share[3] is a plugin for Visual Studio Code and Visual Studio that enables sharing all files of a project loaded in the editor with someone else. In addition to that it enables sharing debugging sessions and ports opened by debugging sessions are forwarded to clients. As with Teletype for Atom files are only persisted on the "host"

2.1.3 Multihack-Brackets

Multihack-Brackets[4] is a plugin for the Brackets editor. It enables sharing an entire folder structure. It requires a server. As of 13.3.2019 it is not possible to verify performance or functionality since joining a session just crashes the brackets editor.

2.1.4 Codeshare

Codeshare[5] is a web based collaborative editor. It is designed for interviews. The editor window offers syntax highlighting for a broad range of programming languages. One shared room always only contains a single file.

2.1.5 Summary

As described in 21 current solutions are mostly implemented as extensions to code editors. Only codeshare is web based because it is based around the use-case of doing interviews. Overall none of the current solutions have any considerations for dealing with an underlying version control system of a project.

2.2 CRDT

In order to update shared objects stored at different sites a "commutative replicated data type" or CRDT is proposed by [8]. The idea is to design the underlying representation or data structure of

[1] https://github.com/atom/teletype/issues/211
[2] https://teletype.atom.io/
[3] https://visualstudio.microsoft.com/de/services/live-share/
[4] https://github.com/multihack/multihack-brackets
[5] https://codeshare.io

Tool	Type	Location	Shared Content
Teletype for Atom	Extension	Host	Individual Files
Visual Studio Live Share	Extension	Host	Project Folder
Multihack Brackets	Extension	Distributed?	Project Folder
Codeshare	Web application	Server	Single File

Table 21: Overview state of the art

edits to a document such that operations are commutative and therefore automatically converge at every copy of the document. But there is a problem with this approach. If there are concurrent insertions at the same position of a document, a global order for the conflicting information has to be established. [8],[5] have a solution to this problem: every site gets a unique siteID and a logical clock or counter. Concurrent inserts are then ordered either by the counter (smaller counter first) or if the counters are identical by the siteID.

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.

2.2.1 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.

2.2.2 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>}`)

2.2.3 Tabellen

Jede Tabelle muss im Fließtext referenziertw werden. Für Tabellen gelten die selben Regeln, wie für Abbildungen (siehe dazu Abschnitt 2.2.2).

Eine Beispiel einer Tabelle ist in Tabelle 22 zu finden:

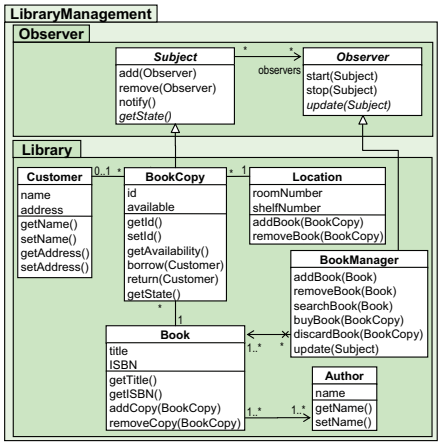

Figure 21: xxx (Quelle zitieren, wenn nicht selbst erstellt)

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

Table 22: xxx (Quelle angeben)

Bitte beachten Sie, dass Tabellen generell so einfach wie möglich gehalten werden sollen. Tabelle 22 dient unter anderem dazu Studierenden zu zeigen, wie Tabellen in LaTeX erstellt werden können und wie Farben verwendet werden.

3 Requirements

3.1 Share the entire project structure

Every user should be able to see all changes to the project. Based on this Issue and the discussion "Current implementation only shares current project. It would be more useful if I could share the whole project so two persons can work on different files."[1] being able to edit the entire project structure is a very important feature.

3.2 Only display changes in the same Git branch

Given that a lot of developers are using prototype branches and a significant number are usign feature branches Git branches are a good indication that a specific problem is being worked on.[7] Therefore only displaying cuncurrent edits on the same edit removes noise of unrelated edits.

3.3 Stage changes by author

Synchronizing changes to all developers introduces a problem: "[...]This means that git only becomes a way to have a backup as all the work is done using P2P! [...]"[2] Possibly unrelated modicications would be bundled into huge commits. In order to mitigate this, changes should be stageable by author.

3.4 Retrieve required information from Git

As Git already contains information about the project and the author, the user should not have to enter this information into the extension again. Instead the extension should, whenever possible, read information (such as the current username) from Git.

3.5 Respect .gitignore

Files explicitly excluded from the version control system via the .gitignore file[3] should not be synchronized with other client as these files might contain automatically generated files that depend on the local system configuration or contain sensitive information.

3.6 Performance

Although "VS Code aims to deliver a stable and performant editor to end users, and misbehaving extensions should not impact the user experience. The Extension Host in VS Code prevents extensions from:

[1] https://github.com/atom/teletype/issues/211
[2] https://github.com/atom/teletype/issues/211#issuecomment-478306010
[3] https://git-scm.com/docs/gitignore

- Impacting startup performance
- Slowing down UI operations
- Modifying the UI

"[4] Performance of the extension should be good enough that typing on two computers with one hand each should be possible without introducing errors or noticable delay if both computers are connected to the same network via ethernet.

Scenario: A person is sitting in front of two computers both with vscode and the extension open. The left hand is on the keyboard of the first computer, the right hand on the keyboard of the second computer. The person should be able to type a sentence without errors being introduced by delays in change synchronization.

3.7 Disable foreign changes

In order to compile source code it is important that the code is free of syntax errors and does not change during compilation. Therefore it should be possible to disable recieving changes from other clients. Sometimes it might even be nessesary to roll back the changes other people have made to the codebase.

3.8 Support bad internet connection

The extension should be able to support the client loosing the internet connection. Even if the code editor is relaunched while offline the changes of other users should still be as they were when the connection was lost. As soon as connectivity is restored, new changes should start to be displayed.

[4] https://code.visualstudio.com/api/advanced-topics/extension-host

4 User stories

As a programmer I want to have access the entire project structure when using a code editor.

As a programmer I want clean seperation between branches and do not want to see changes to other branches when using a version control system.

As a programmer I want to be able to stage changes that I made to Git.

As a project manager I want to see who made a specific modicication to a project in the version control system.

As a programmer I do not want to configure a second version control system when I already provided this information to Git.

As a programmer I do not want files covered by my .gitignore configuration to be shared with others.

As a programmer I want to be able to edit files with up to 4 people at a time. A file can have up to 30.000 characters.

5 Implementation

5.1 VS Code Extension API

VS Code runs extension in a seperate process and provides an async Javascript API. The examples provided in the VS Code examples[1] are mostly written in Typescript[2] and all the Interfaces have type definitions for Typescript. Therefore the Extension will use Typescript as well.

5.1.1 Used API functions

```
1 vscode.workspace.onDidChangeTextDocument()
```

"An event that is emitted when a text document is changed. This usually happens when the contents changes but also when other things like the dirty-state changes."[3]

```
1 vscode.workspace.onDidOpenTextDocument()
```

"An event that is emitted when a text document is opened or when the language id of a text document has been changed."[4]

[1] https://github.com/Microsoft/vscode-extension-samples
[2] https://www.typescriptlang.org/
[3] https://code.visualstudio.com/api/references/vscode-api#workspace
[4] https://code.visualstudio.com/api/references/vscode-api#workspace

6 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.

7 Hinweise zur Literatur

7.1 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

7.2 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 [3] ...
In [9] zeigen Schanes et al. wie ... Weitere Informationen können in [4] von Oasis entnommen werden.

Wir empfehlen JabRef, um die Literaturdatenbank zu verwalten.

8 Algorithmen und Quellcode

8.1 Beispiele für Quellcode

Beispiel eines Quellcodes ist im Quellcode 8.1 zu finden.

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

Listing 8.1: Short code

8.2 Beispiele für Algorithmen

Algorithmus 8.1 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 8.1: Sample algorithm

9 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.

10 Zusammenfassung und Ausblick

Bibliography

References

[1] Brian de Alwis and Jonathan Sillito. „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] Loïck Briot, Pascal Urso, and Marc Shapiro. „High Responsiveness for Group Editing CRDTs“. In: *ACM New York, NY, USA ©2016* (2016).

[3] 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).

[5] Gérald Oster et al. „Data consistency for P2P collaborative editing“. In: *ACM New York, NY, USA ©2006* (2006).

[6] Gérald Oster et al. „Supporting String-Wise Operations and Selective Undo for Peer-to-Peer Group Editing“. In: *ACM New York, NY, USA ©2006* (2014).

[7] Shaun Phillips, Jonathan Sillito, and Rob Walker. „Branching and merging: an investigation into current version control practices“. In: *ACM New York, NY, USA ©2011* (2011), p. 12.

[8] Nuno Preguica et al. „A Commutative Replicated Data Type for Cooperative Editing“. In: *IEEE* (2009).

[9] 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

[4] 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, ...