

# Session 2: Versionskontrolle mit git und GitHub

## Fortgeschrittene Programmierung (Java 2)

Nils Reiter

`nils.reiter@uni-koeln.de`

12. April 2023

# Introduction

- ▶ Versioning of source code
- ▶ Differences between versions
- ▶ Maintaining several branches in parallel

## Introduction

- ▶ Versioning of source code
- ▶ Differences between versions
- ▶ Maintaining several branches in parallel

### Why is this useful?

Generally: Dealing with complexity!

## Introduction

- ▶ Versioning of source code
- ▶ Differences between versions
- ▶ Maintaining several branches in parallel

### Why is this useful?

Generally: Dealing with complexity!

- ▶ Programming projects quickly become massive
  - ▶ Windows 2000: 28mio LoC (ca. 930k standard pages)
  - ▶ CorefAnnotator: 27k LoC (ca. 770 standard pages)
- ▶ Large teams
  - ▶ working on the same project
  - ▶ over a long time  $\Rightarrow$  don't rely on human memory!

## Introduction

- ▶ Versioning of source code
- ▶ Differences between versions
- ▶ Maintaining several branches in parallel

### Why is this useful?

Generally: Dealing with complexity!

- ▶ Programming projects quickly become massive
  - ▶ Windows 2000: 28mio LoC (ca. 930k standard pages)
  - ▶ CorefAnnotator: 27k LoC (ca. 770 standard pages)
- ▶ Large teams
  - ▶ working on the same project
  - ▶ over a long time  $\Rightarrow$  don't rely on human memory!
- ▶ A single conceptual change often distributed over many files (e.g., class rename)

## CorefAnnotator

The screenshot shows the CorefAnnotator application interface. The main window displays a text document with several paragraphs of text. The text is annotated with coreference links (underlines) and color-coded tags (e.g., **DER WIRT.**, **WAITWELL.**, **SIR WILLIAM.**). The sidebar on the right shows a coreference graph with nodes and edges, representing the relationships between the annotated entities in the text.

**Zweiter Auftritt**

*Der Wirt. Sir William Sampson. Waitwell.*

**DER WIRT.**  
So früh, meine Herren, so früh? Willkommen! willkommen  
Waitwell! Ihr seid ohne Zweifel die Nacht gefahren? Ist das der Herr, von dem du gestern mit mir gesprochen hast?

**WAITWELL.**  
Ja, er ist es, und ich hoffe, daß du abgeredeter Maßen --

**DER WIRT.**  
Gnädiger Herr, ich bin ganz zu Ihren Diensten. Was liegt mir daran, ob ich es weiß, oder nicht, was Sie für eine Ursache hierher führt, und warum Sie bei mir im Verborgnen sein wollen? Ein Wirt nimmt sein Geld, und läßt seine Gäste machen, was ihnen gut dünkt. Waitwell hat mir zwar gesagt, daß Sie den fremden Herrn, der sich seit einigen Wochen mit seinem jungen Weibchen bei mir aufhält, ein wenig beobachten wollen. Aber ich hoffe, daß Sie ihm keinen Verdruß verursachen werden. Sie würden mein Haus in einen übeln Ruf bringen, und gewisse Leute würden sich scheuen, bei mir abzutreten. Unser einer muß von allen Sorten Menschen leben. --

**SIR WILLIAM.**  
Besorget nichts; führt mich nur in das Zimmer, das Waitwell für mich bestellt hat. Ich komme aus rechtschaffnen Absichten hierher.

**DER WIRT.**  
Ich mag Ihre Geheimnisse nicht wissen, gnädiger Herr! Die Neugierde ist mein Fehler gar nicht. Ich hätte es, zum Exempel, längst erfahren können, wer der fremde Herr ist, auf den Sie Acht geben wollen; aber ich mag nicht. So viel habe ich wohl herausgebracht, daß er mit dem Frauenzimmer muß durchgegangen sein. Das gute Weibchen, oder was sie ist! sie bleibt den ganzen Tag in ihrer Stube eingeschlossen und weint.

CorefAnnotator 1.0.0-SNAPSHOT

Coreference Graph (Right Sidebar):

- böse\_leute (5)
- die\_liebe\_welche\_mir\_forschen\_half
- einbildungen (5)
- juwelen (5)
- tränen\_marwood (5)
- vorwürfe (5)
- DER BEDIENTE (4)
- bilder (4)
- der\_mensch (4)
- diese\_verruchte\_hand (4)
- erbschaft (4)
- feile\_buhlerinnen (4)
- kleinigkeit (4)
- miene\_marwood (4)
- neue\_freunde (4)
- ruf\_marwood (4)
- schönheit\_marwood (4)
- schöpfer (4)
- spieler\_und\_landstreicher (4)
- verbrechen (4)
- Ein Wirt (3)
  - Ein Wirt
  - sein
  - seine
- dank (3)
- der\_welcher\_tugendhaft\_sein\_soll (3)
- die\_liebe (3)
- diese\_schimpfliche\_erbschaft (3)
- ein\_mittel (3)
- eine\_lüge (3)
- eine\_sprache (3)
- euch\_unmenschliche\_tyramnen\_unse
- feier (3)
- haus\_sara (3)
- himmel (3)
- ich\_würde\_ihr\_doch\_vergeben (3)
- ihr\_name (3)
- ihre\_tugend (2)

# CorefAnnotator

- ▶ Typical research software
- ▶ <https://github.com/nilsreiter/CorefAnnotator/>
- ▶ Annotation tool for coreference chains  
(and various other things)
- ▶ Open source, Apache License
- ▶ Version 1.0: February 12, 2018
- ▶ Version 2.0: May 25, 2021

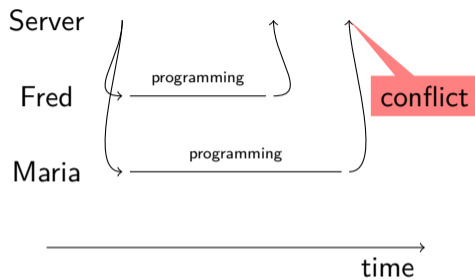
Version	LoC
1.0	13 087
1.5	22 034
1.10	33 436
1.15	38 259
2.0	42 910
2.1	43 021

## Collaboration Options

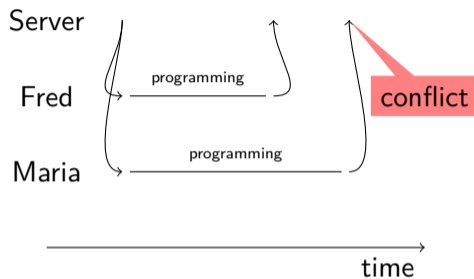
The slide features a light gray grid pattern on a white background. The title 'Collaboration Options' is positioned in the top-left corner in a dark blue font. The rest of the slide is empty, providing a space for content.



# Situations



# Situations



## Conflict resolution options

- ▶ Ignore, let Maria overwrite Freds code (this is bad!)
- ▶ Create a second copy (this is what Dropbox does)
- ▶ Force Maria to *explicitly* merge the code: Look at both versions and decide what should remain

# Software for Version Control


- ▶ Very old
  - ▶ CVS (concurrent versioning system)
  - ▶ Rarely used today
- ▶ Old
  - ▶ SVN (subversion)
  - ▶ Sometimes used
- ▶ State of the art
  - ▶ `git`
- ▶ More solutions are available commercially

# git

- ▶ Developed by the Linux kernel developers
- ▶ Open source – <https://git-scm.com>
  - ▶ I.e., you can download the source code of git
- ▶ Distributed: No central server required
  - ▶ ...but it's still useful to have one
- ▶ Fast
- ▶ Data assurance
  - ▶ Checksums to make sure you get out what you put in

github.com/git/git/blob/e83c5163316f89bfbde7d9ab23ca2e25

e83c516331 git / README Go to file



 **VanTudor** Initial revision of "git", the information manager from hell Latest commit e83c516 on 8 Apr 2005 History

1 contributor

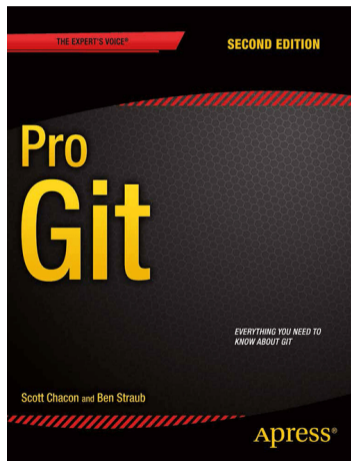
168 lines (135 sloc) | 8.2 KB Raw Blame

```
1
2     GIT - the stupid content tracker
3
4 "git" can mean anything, depending on your mood.
5
6 - random three-letter combination that is pronounceable, and not
7 actually used by any common UNIX command. The fact that it is a
8 mispronunciation of "get" may or may not be relevant.
9 - stupid. contemptible and despicable. simple. Take your pick from the
10 dictionary of slang.
11 - "global information tracker": you're in a good mood, and it actually
12 works for you. Angels sing, and a light suddenly fills the room.
13 - "goddamn idiotic truckload of sh*t": when it breaks
14
15 This is a stupid (but extremely fast) directory content manager. It
16 doesn't do a whole lot, but what it _does_ do is track directory
17 contents efficiently.
18
19 There are two object abstractions: the "object database", and the
20 "current directory cache".
21
```

## git vs. GitHub vs. GitLab

- ▶ git is an open source software
  - ▶ <https://git-scm.com>
  - ▶ Integrated into many other tools – e.g., Eclipse
- ▶ GitHub is a (commercial) web platform 
  - ▶ Recently bought by Microsoft
  - ▶ GitHub provides a central server for git repositories *and* additional services (wiki, ticket system, ...)
  - ▶ <https://github.com>
- ▶ GitLab is an open source software 
  - ▶ Provides a central server that you can install on your own server (e.g., at the CCeH)
  - ▶ “GitHub for your own server”
  - ▶ <https://about.gitlab.com>

# Reading



Scott Chacon and Ben Straub: “Pro Git”. 2nd edition.  
Apress, 2014.

<https://git-scm.com/book/en/v2>

## Table of Contents

1. Getting Started
2. Git Basics
3. Git Branching
4. Git on the Server
5. Distributed Git
6. ...



## Section 2

How does git work?



## Commit

- ▶ Commit: One version of an entire directory (including subdirectories)
- ▶ Creating commits is the central activity we do
- ▶ Each commit knows its predecessor

# Commit

- ▶ Commit: One version of an entire directory (including subdirectories)
- ▶ Creating commits is the central activity we do
- ▶ Each commit knows its predecessor
- ▶ Each commit is identified by a hash value:  
0eabb4bfef80be2af18255dc19301b989da1f1a3
- ▶ A commit can include changes in multiple files
- ▶ Registering your changes is a two-step process
  1. Put in staging area
  2. Commit everything in staging area
- ▶ Commit and version are related, but not the same concepts

# Lifecycle

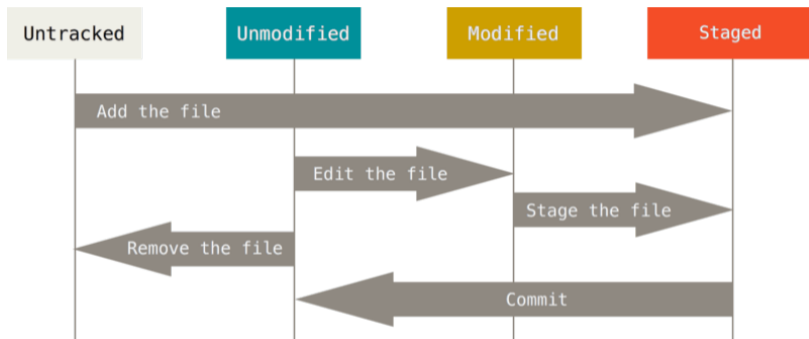


Figure: The lifecycle of the status of your files (Chacon/Straub: Pro Git)

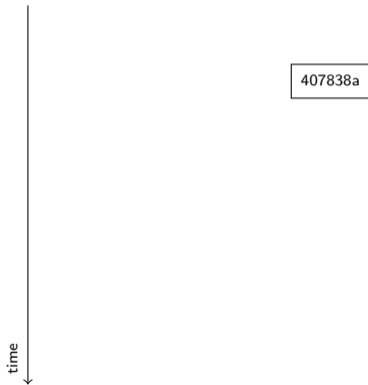
# Workflow

1. (Pull changes from others)
2. Edit/add files
3. Put files in staging area
  - ▶ `git add <FILENAME>`
  - ▶ `git remove <FILENAME>`
4. Commit all files in staging area
  - ▶ Provide a useful description
  - ▶ `git commit -m "comment"`
5. (Push to others)

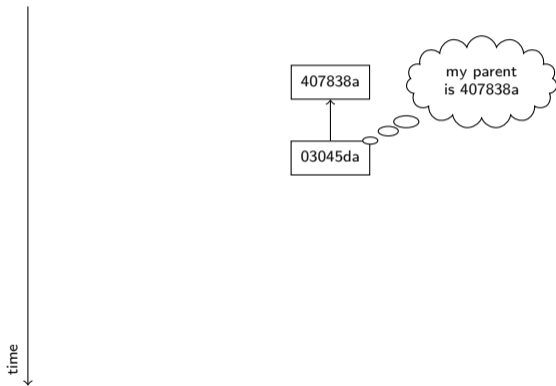
# Branching

- ▶ Maintaining multiple branches is often useful
- ▶ At each time, a single branch is active
  - ▶ By default: `master`
- ▶ Switch to an existing branch
  - ▶ `git checkout <BRANCHNAME>`
  - ▶ To create a new branch, add the option `-b`:
    - ▶ `git checkout -b <BRANCHNAME>`

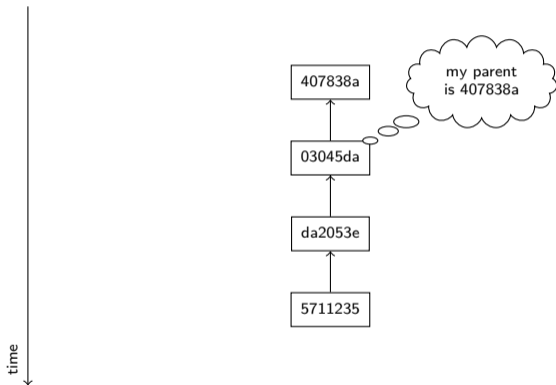
## Branching and committing results in a tree



## Branching and committing results in a tree

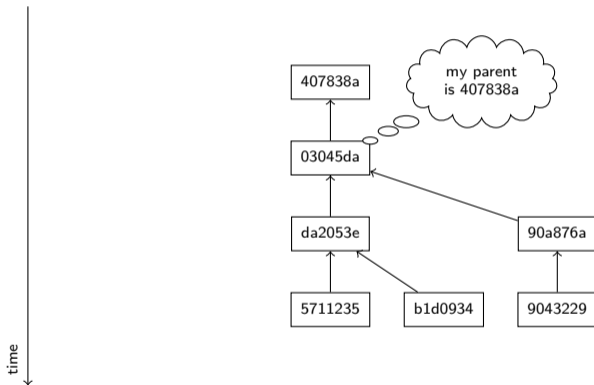


## Branching and committing results in a tree

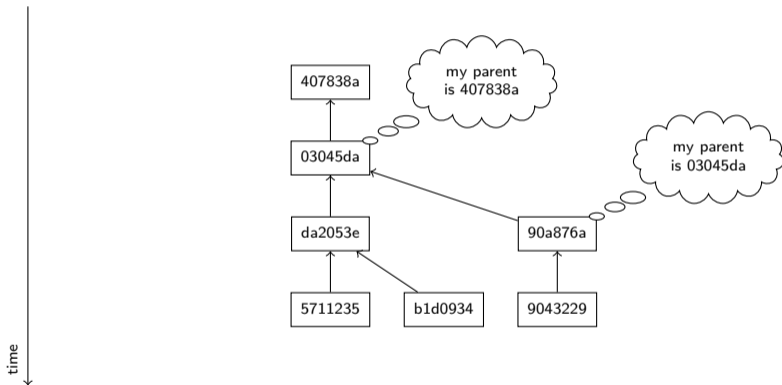




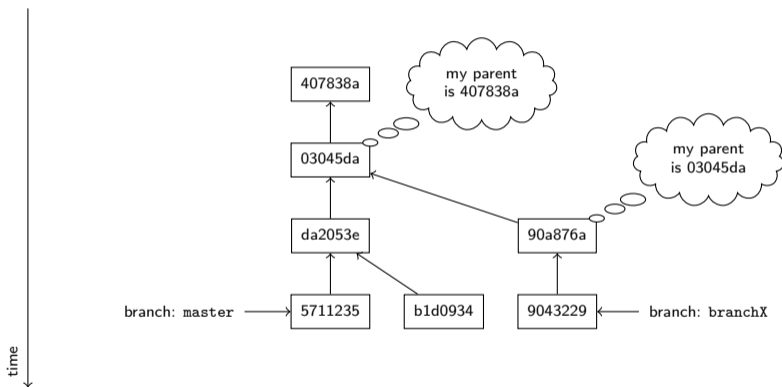
## Branching and committing results in a tree



## Branching and committing results in a tree



# Branching and committing results in a tree



demo

## What do we put under version control?

### All variants of plain text files

- ▶ source code (python, java, perl, c, ...)
- ▶ texts (plain, latex, markdown)
- ▶ primary data (xml, csv)
  - ▶ but beware of large files
- ▶ vector graphics (svg)

## What do we put under version control?

### All variants of plain text files

- ▶ source code (python, java, perl, c, ...)
- ▶ texts (plain, latex, markdown)
- ▶ primary data (xml, csv)
  - ▶ but beware of large files
- ▶ vector graphics (svg)

### Do not put these in version control

- ▶ word documents, pdf files
- ▶ images (jpg, png)
- ▶ compiled code (executables)

## What do we put under version control?

### All variants of plain text files

- ▶ source code (python, java, perl, c, ...)
- ▶ texts (plain, latex, markdown)
- ▶ primary data (xml, csv)
  - ▶ but beware of large files
- ▶ vector graphics (svg)

### Do not put these in version control

- ▶ word documents, pdf files
- ▶ images (jpg, png)
- ▶ compiled code (executables)
- ▶ Exceptions apply

## Repository vs. working copy

- ▶ The git repository keeps track of *all* past versions and branches
- ▶ The working copy is set to one specific version (designated by HEAD)
- ▶ `git checkout REFNAME`
  - ▶ REFNAME can be a branch or revision hash (or tag)
- ▶ Checking out moves the HEAD pointer to another revision
  - ▶ The HEAD pointer always points to the revision that's active in your working copy



## Remotes

- ▶ Git repositories can be associated with *remote* repositories
  - ▶ Remote repositories are usually on a different computer (e.g., GitHub)

# Remotes

- ▶ Git repositories can be associated with *remote* repositories
  - ▶ Remote repositories are usually on a different computer (e.g., GitHub)
- ▶ A repository needs to be synchronized with its remote manually:
  - ▶ `git clone REPOURL`: Create a local copy of the repository, setting REPOURL as 'origin' remote
    - ▶ Usually, used only once
  - ▶ `git push`: Transfers the commits on the local branch to the same branch on the remote
  - ▶ `git pull`: Transfers the commits on the remote branch to the local branch

## Useful commands

```
git status
```

Shows the status of the current working copy

- ▶ Changed files
- ▶ Files in the staging area
- ▶ The current branch

## Useful commands

### `git status`

Shows the status of the current working copy

- ▶ Changed files
- ▶ Files in the staging area
- ▶ The current branch

### `git log`

Shows information about current and past commits

Useful options:

- `--oneline` Each commit is shown on a single line
- `--graph` Information is rendered visually
- `--all` Shows information about all branches

## On GUIs

Git has a complex task and is a complex piece of software

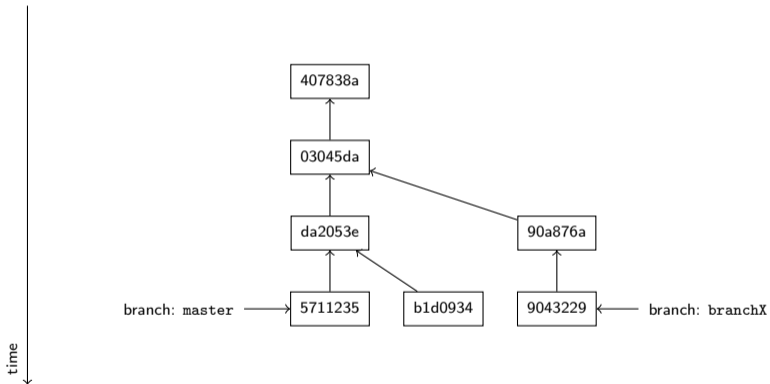
- ▶ Graphical user interfaces do exist and make some tasks easier
- ▶ In this class: command line and Eclipse
  - ▶ git commands can be found in the context menu under “Team”

## On GUIs

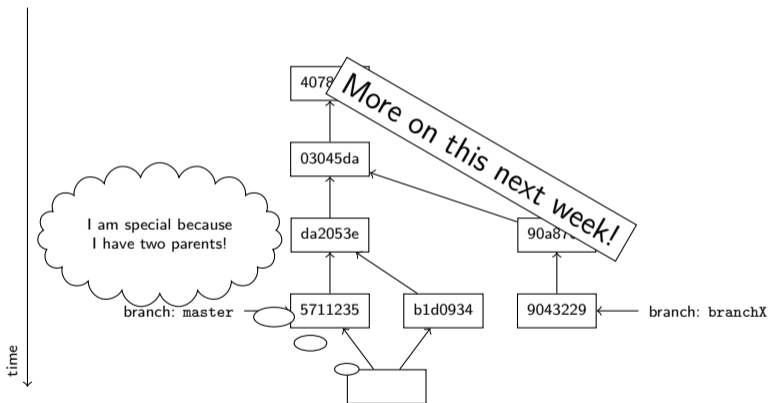
Git has a complex task and is a complex piece of software

- ▶ Graphical user interfaces do exist and make some tasks easier
- ▶ In this class: command line and Eclipse
  - ▶ git commands can be found in the context menu under “Team”
- ▶ Other tools (sometimes better visualizations!)
  - ▶ SourceTree (Win/Mac): <https://www.sourcetreeapp.com>
    - ▶ Needs a registration with BitBucket (similar to GitHub), but free
  - ▶ GitKraken (Win/Mac/Lin): <https://www.gitkraken.com>
    - ▶ Free for open source projects
- ▶ More can be found here:  
<https://git-scm.com/downloads/guis/>

# Sneak Peak: Merging



# Sneak Peak: Merging





# Exercise



`https://github.com/idh-cologne-java-2-summer-2023/exercise-02`