Session 2: Versionskontrolle mit git und GitHub

Fortgeschrittene Programmierung (Java 2)

Nils Reiter
nils.reiter@uni-koeln.de

12. April 2023

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

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

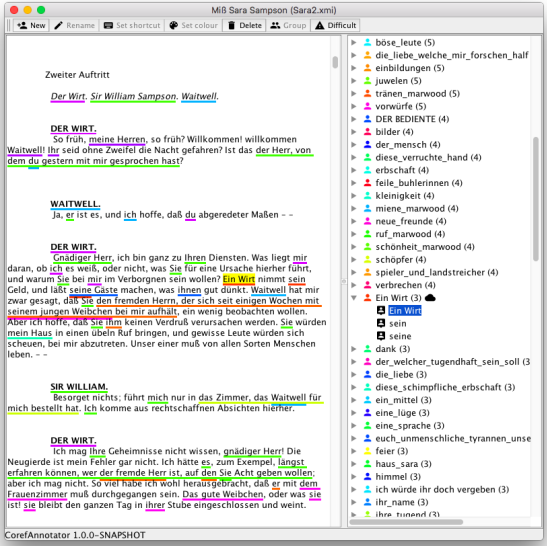
- Versioning of source code
- Differences between versions
- Maintaining several branches in parallel

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



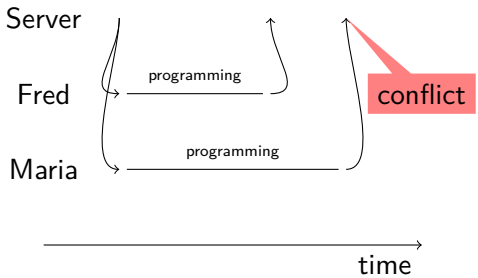
CorefAnnotator

- 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

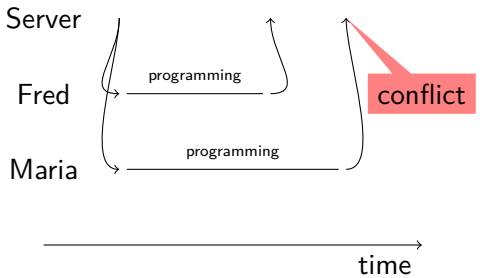
Version	LoC
1.0	13 087
1.5	22 034
1.10	33 436
1.15	38 259
2.0	42 910
2.1	43 021

Collaboration Options

Situations



Situations



Conflict resolution options

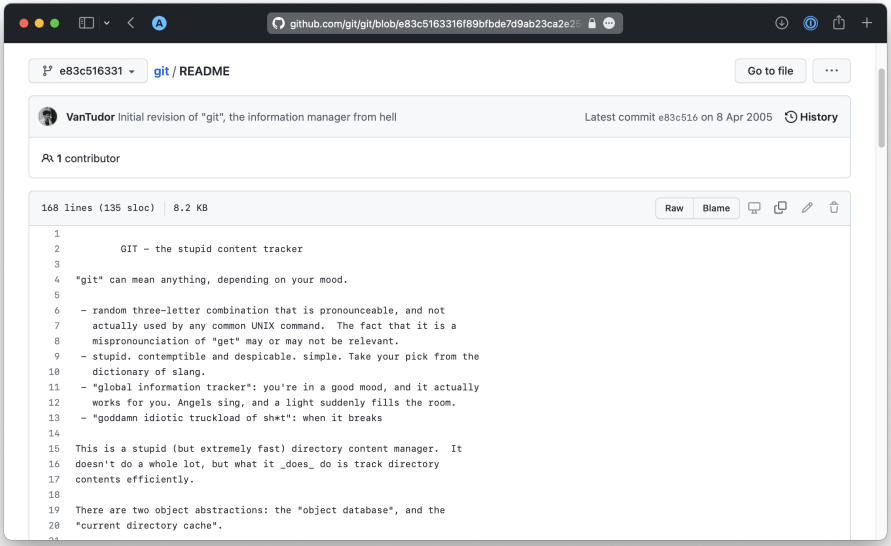
- Ignore, let Maria overwrite Freds code (this is bad!)
- Create a second copy (this is what Dropbox does)
- Force Maria to *explicitly* merge the code: Look at both versions and decide what should remain

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

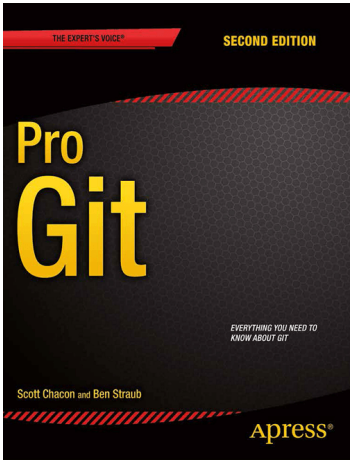
- Developed by the Linux kernel developers
- Open source – `https://git-scm.com`
  - I.e., you can download the source code of git
- Distributed: No central server required
  - ...but it's still useful to have one
- Fast
- Data assurance
  - Checksums to make sure you get out what you put in


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

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

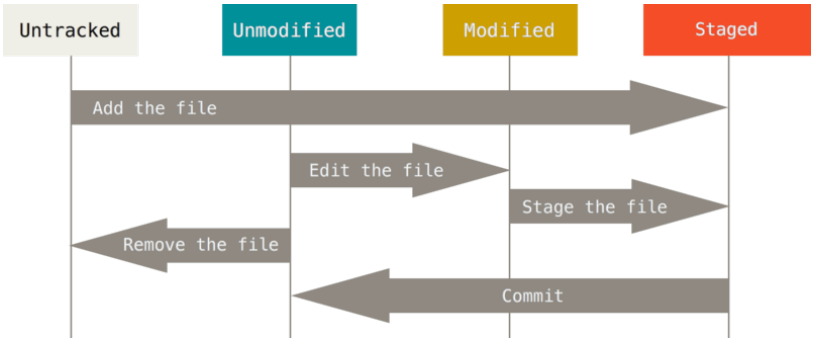


Figure: The lifecycle of the status of your files (Chacon/Straub: Pro Git)

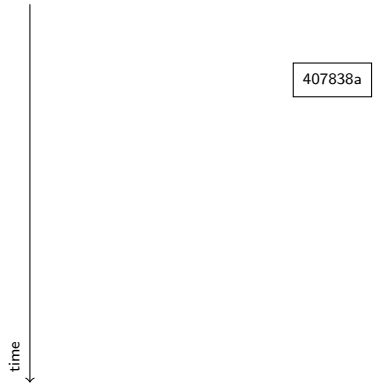
Workflow

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)

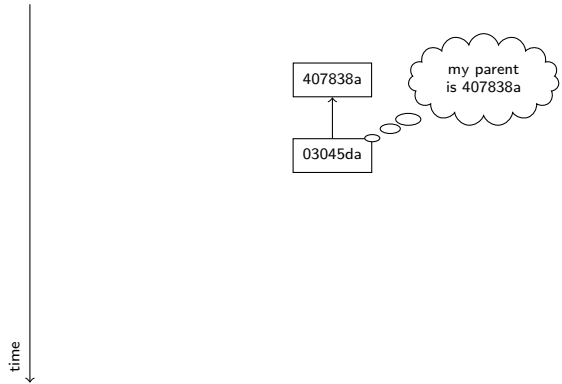
Branching

- 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>`

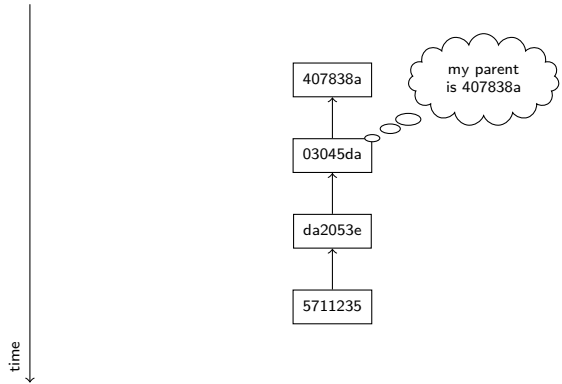
Branching and committing results in a tree



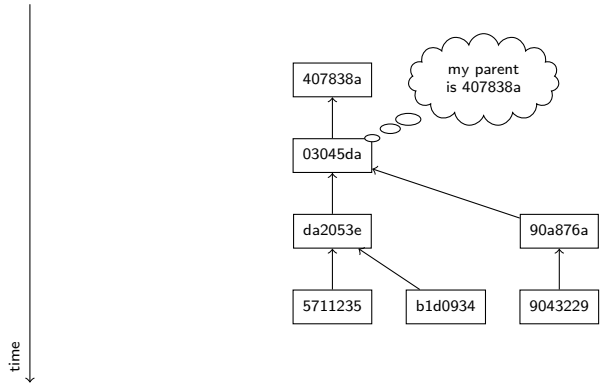
Branching and committing results in a tree



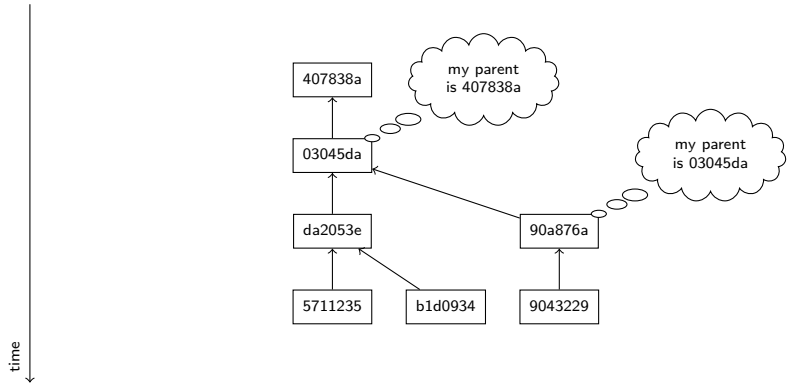
Branching and committing results in a tree



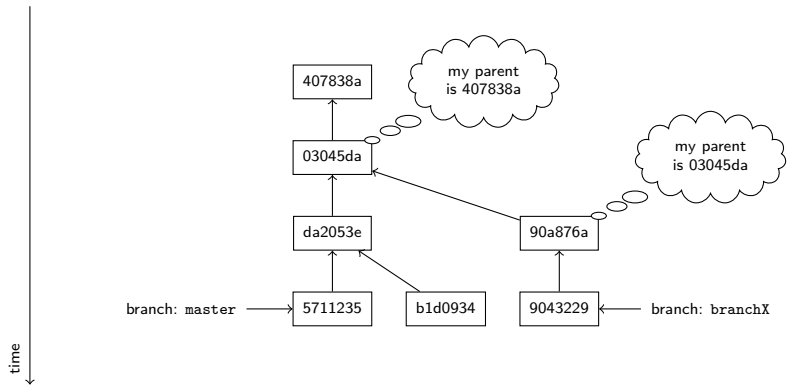
Branching and committing results in a tree



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

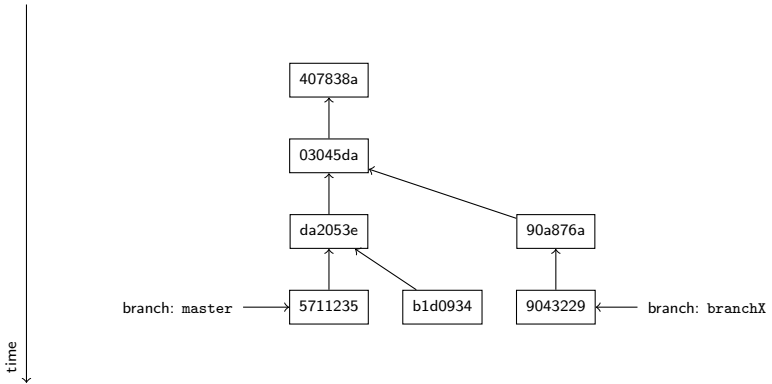
- Graphical user interfaces do exist and make some tasks easier
- In this class: command line and Eclipse
  - git commands can be found in the context menu under “Team”

On GUIs

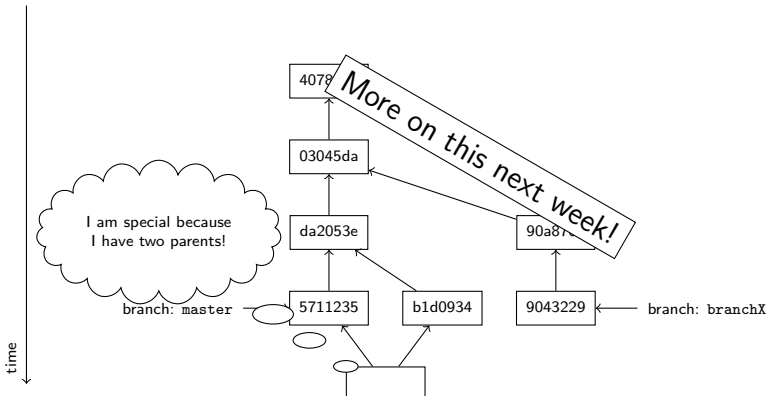
Git has a complex task and is a complex piece of software

- 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