Gettin' the hang of git

Version control and why git matters

Sinthujan Thanabalasingam

Worüber Wir sprechen

- Kollaborative Softwareentwicklung
- Probleme
- git – Was und wieso?
- Basic Workflow (to git gud)
- Hinweise zu git im Praktikum

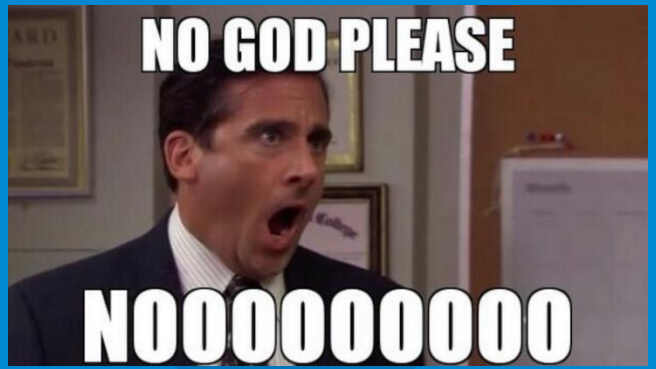
Software Engineering im Team

- komplexe Projekte setzen große Teams voraus
- moderne SE ist kollaborativ und verteilt
- Arbeitsteilung zentraler Aspekt
- Integration aller Einzelteile zu einem Ganzen

→ Wie verwaltet man am besten die Codebase?

Easy!

USB-Sticks! Einfach Projekt auf einem Stick speichern und dem Kollegen geben!



Probleme der “einfachen” Lösung

- Arbeitsteilung im Team?
- Datenträgerfehler und andere Katastrophen?
- Rückgängigmachen von Änderungen?
- Versionierung?
- Rettung aus Hass und Verzweiflung?
- Was ist gerade die gemeinsame Arbeitsbasis?

Version Control System (VCS)

- Tracking von verteilten Dokumenten und Änderungen
- Source code management
- Wiederherstellung vorheriger Zustände
- Automatische Integration (merging) verschiedener Änderungen
- essentiell für die Organisation von Multi-Developer Projekten

git to the rescue

- verteiltes VCS
- kooperatives Arbeiten an einem Projekt
- Verwaltung in Form von Repositories
- webbasierte Hosting Services wie GitHub oder Bitbucket
- für Linux, Windows und macOS verfügbar

The guts of git

- jeder Entwickler arbeitet an lokaler Kopie des Projekts (local repository)
- Änderungen werden zusammengefasst und verwaltet (commit)
- commits werden auf remote repositories publiziert
- Entwickler können Änderungen anderer Entwickler über das remote repository einpflegen

How to get git

Debian/Ubuntu und WSL

```
sudo apt-get install git-all
```

Windows

- git bash: https://git-scm.com/download/windows
- WSL (Windows Subsystem for Linux)

macOS

https://git-scm.com/download/mac

git – Basic Workflow

Jedes Verzeichnis kann mit Hilfe von git verwaltet werden.

```
git init
```

- initialisiert ein Repository im Arbeitsverzeichnis
- jetzt können Dateien zum Index hinzugefügt werden
- git verwaltet alle im Index befindlichen Dateien

git init – result

```
Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository
$ git init
Initialized empty Git repository in D:/Development/myRepository/.git/

Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository (master)
$ ls -lha
total 8.0K
drwxr-xr-x 1 Sinthu 197609 0 Feb 18 17:39 ./
drwxr-xr-x 1 Sinthu 197609 0 Feb 18 17:38 ../
drwxr-xr-x 1 Sinthu 197609 0 Feb 18 17:39 .git/

Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository (master)
$
```

versteckter Ordner für git

master branch (dazu gleich mehr)

git – Basic Workflow

- nach dem init kann am Projekt gearbeitet werden
- Erstellen/Updaten/Löschen von Dateien
- git registriert Änderungen im Arbeitsverzeichnis
- Änderungen können dann veröffentlicht werden
- add, commit, branch, push und pull sind elementare Befehle

Status des Repository

```
git status
```

- Zusammenfassung der Änderungen seit letztem commit
- Listet neue, gelöschte und geänderte Dateien auf
- unterstützt bei Entscheidung,
  welche Änderungen in einem commit eingepflegt
  werden sollen

```
Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   sourcecode.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        newfile.py

no changes added to commit (use "git add" and/or "git commit -a")
```

neue Datei
newfile.py

Änderungen an
sourcecode.py

git – Basic Workflow

Das Hinzufügen von Dateien zum Index wird „staging“ genannt.

```
git add <filename>
```

fügt Datei namens <filename> hinzu

```
git add *.<extension>
```

fügt alle Dateien mit der Dateiendung <extension> hinzu

```
git add .
```

fügt alle Dateien hinzu

```
Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository (master)
$ git add newfile.py

Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   newfile.py

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   sourcecode.py
```

Hinzufügen von newfile.py zum Index

neue Datei newfile.py

bereits im Index vorhanden, aber modifiziert

git commit

„Ein Commit beschreibt eine Menge von Änderungen an Ressourcen eines Repository.“

git commit

```
git commit -m "commit message"
```

- fasst Änderungen zu einem commit zusammen
- Änderungen werden lokal ins VCS eingepflegt
- commit message
  - sollte sinnvoller Beschreibung der Änderungen entsprechen
  - ist für andere Developer im Team sichtbar
  - “changed stuff”, “bug fix“, “more work“, “minor changes“ oder “pls kill me“ sind keine guten messages…

```
Sinthu@TrentsThinkpad MINGW64 /c/Development/myRepository (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   modified.py

no changes added to commit (use "git add" and/or "git commit -a")

Sinthu@TrentsThinkpad MINGW64 /c/Development/myRepository (master)
$ git add modified.py

Sinthu@TrentsThinkpad MINGW64 /c/Development/myRepository (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   modified.py


Sinthu@TrentsThinkpad MINGW64 /c/Development/myRepository (master)
$ git commit -m "fixed issue #25. introduced unicorns"
[master b26661f] fixed issue #25. introduced unicorns
 1 file changed, 4 insertions(+)
```

git commit

- Jeder commit wird durch einen Hashwert identifiziert
- Repositories können mit Hilfe der Hashes auf frühere Zustände zurückgesetzt werden
- Commits geschehen zunächst lokal, beeinflussen also nur das eigene Arbeitsverzeichnis

Now, you might ask yourself…

Wie wird das Ganze kollaborativ?

Wie publiziert man seine Änderungen?

Und vor allem: wohin?

git remote got you covered

```
git remote add <alias> <url>
```

- verknüpft local repository mit einem remote repository
- <alias>: Alias, unter dem der remote lokal bekannt ist
- <url>: URL des remote repository

```
git remote -v
```

- zeigt alle verfügbaren remote repositories an

```
Sinthu@Sinthu-pc MINGW64 ~/Desktop/git slides (master)
$ git remote -v

Sinthu@Sinthu-pc MINGW64 ~/Desktop/git slides (master)
$ git remote add origin https://github.com/edisontrent1337/gitslides.git

Sinthu@Sinthu-pc MINGW64 ~/Desktop/git slides (master)
$ git remote -v
origin  https://github.com/edisontrent1337/gitslides.git (fetch)
origin  https://github.com/edisontrent1337/gitslides.git (push)
```

alias origin für die lange URL

fetch und push für read und write vom/auf remote

Share your bugs with the world!

```
git push <alias> <branch>
```

- pusht alle nicht veröffentlichten commits auf den Branch <branch> des remote repository mit dem Alias <alias>
- origin ist lokaler Standard-Alias für remote repository
- alias vereinfacht pushen, da volle URL eines remote repository nicht angegeben werden muss

```
Sinthu@Sinthu-pc MINGW64 ~/Desktop/git slides (master)
$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 94.24 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/edisontrent1337/gitslides.git
   c36ffce..f7d7803  master -> master
```

Commits on Dec 19, 2017

changed main menu
edisontrent1337 committed on 19 Dec 2017 — 235aeab

Commits on Dec 13, 2017

fixed #53 and #59
edisontrent1337 committed on 13 Dec 2017 — 1d04f88

fixed #40
edisontrent1337 committed on 13 Dec 2017 — 6a85698

Commits on Dec 10, 2017

fixed button issue with a hack
edisontrent1337 committed on 10 Dec 2017 — 2f57f7d

added image button support to all views.
edisontrent1337 committed on 10 Dec 2017 — b1501f7

householding before changes
edisontrent1337 committed on 10 Dec 2017 — 929f67c

Commits on Dec 9, 2017

further bug fixes: fixed zoom related issues with distance markers.
edisontrent1337 committed on 9 Dec 2017 — 31eec8a

fixed some bugs regarding camera zoom
edisontrent1337 committed on 9 Dec 2017 — a5d0745

fixed camera code
edisontrent1337 committed on 9 Dec 2017 — 93def8e

Branches

- stellen separate History eines Zustands des Repositories dar
- separate Sichten auf das Repository
- geeignet, um Features unabhängig von anderen Branches und Developern zu entwickeln
- master ist der default branch

git branch

```
git branch <name>
```

- erstellt neuen branch namens <name>
- branch ist nach Erstellung exakte Kopie des current branch (beide branches sind even)

```
git checkout <branch>
```

- eigentlicher Wechsel auf den branch <branch>
- Änderungen betreffen ab jetzt nur neuen branch

Dude check ‘git checkout’ out

- ermöglicht das Wechseln zwischen Sichten auf verschiedene Objekte
- git checkout ist für einzelne files, commits und branches möglich

```
git checkout -b <branch>
```

- Kurzform: erstellt einen branch und wechselt auf diesen

Branching visualisiert