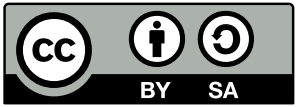
L02 Source Code Management

Markus Raab

Institute of Information Systems Engineering, TU Wien



Local

Learning Outcomes

After successful completion of L02
students will be able to

- use source code management in FLOSS context
- review source code in FLOSS context

Git

- initiated by Linus Torvalds
- content-addressable filesystem or object store
- low-level tools allow to build object graph
- porcelain commands for source code management on top

> *"Smart data structures and dumb code works a lot better than the other way around."*
> *– Eric S. Raymond*

Elektra has KeySet as datastructure.

Tool Suite Git

- common functionality, e.g., `--help` opens man pages
- `git` is a wrapper calling other subcommands
- e.g., `/usr/lib/git-core/git-bisect` is a shellscript

As in Elektra's kdb tool suite.

Rebase vs. Merge

- rebase rewrites commits
- rebase to be avoided if others already pulled
- merge creates merge commit
- merge is more often conflict-free

Daily Work

- stash
- write your own git subcommands
- aliases via config
- ssh keys

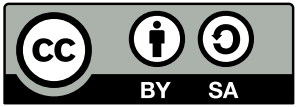
Task

Do you agree with that list? Discuss your experiences.

L02 Source Code Management

Markus Raab

Institute of Information Systems Engineering, TU Wien



Decentralized

Workflows

- patches by email
- create your fork and do pull requests via web

Finding

Decide for one standard workflow for your FLOSS initiative.

Issue Tracker Integration

- `@mention`
- closes/fixes `#issue`

Finding

Prefer having all information directly in source code or git history.

Before Pull Requests

- Rebase to current master.
- If preferred by you: Squash unnecessary commits.
- Write a line in release notes
- Look through commit message.
- Look at what your Pull Request would change

Finding

Prefer having all information directly in source code or git history.

Signing

GPG-sign vs. signoff:

- Commits
- Tags

Finding

sign commits or tags of releases

Best Practices

- always work on branches in your own fork
- separate different things in different commits
- always pull before working
- avoid `--force` push, never `--force` push on master
- `--rebase --autostash`
- rebase+squash only before pushing

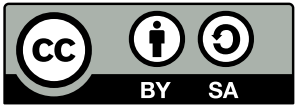
Task

Do you agree with that list? Discuss your experiences.

L02 Source Code Management

Markus Raab

Institute of Information Systems Engineering, TU Wien



Reviews

Introduction

> *"Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone."*
> *– Eric S. Raymond*

Linus's law:

> *"given enough eyeballs, all bugs are shallow"*

Who Reviews?

- experienced programmers
- maintainers
- “extern programmers”
- everyone who has time and concentration

How to Review?

- reading the code
- as little review criteria as possible
- standard criteria in `.github/PULL_REQUEST_TEMPLATE.md`
- only important comments (avoid nitpicking)
- if automated, check if the check was running

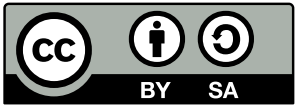
Goals

- Testing with source-code awareness.
- Review everything.
- Have enough “core developers” and reviewers.
- Netiquette same as in issue tracker.

L02 Source Code Management

Markus Raab

Institute of Information Systems Engineering, TU Wien



Meeting

L03 Development Tools

Completely reworked.

[1] Markus Raab and Gergö Barany. Introducing context awareness in unmodified, context-unaware software. In *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE,*, pages 218–225. INSTICC, ScitePress, 2017. ISBN 978-989-758-250-9. doi: 10.5220/0006326602180225.