L09 Architecture

Markus Raab

Institute of Information Systems Engineering, TU Wien

This work is licensed under a Creative Commons *"Attribution-ShareAlike 4.0 International"* license.

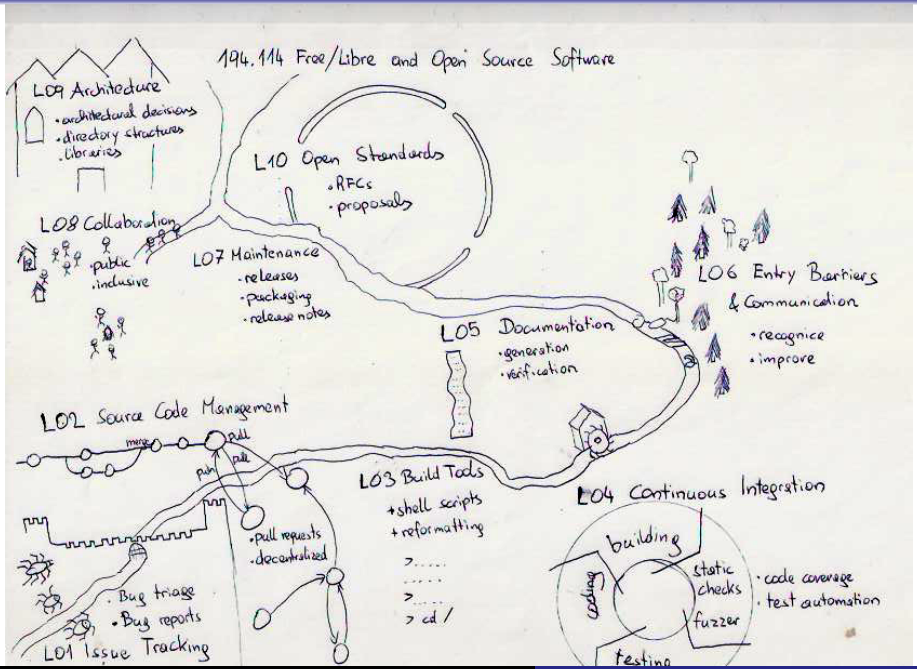
Directory Structure

1. Directory Structure
2. Software Architecture
3. Architectural Decisions
4. Meeting
   - Recapitulation
   - Assignments
   - Preview

Learning Outcomes

After successful completion of L09 Architecture students will be able to

- reproduce chosen FLOSS software architectures.


194.114 Free/Libre and Open Source Software
L09 Architecture
•architectural decisions
•directory structures
•libraries
L10 Open Standards
•RFCs
•proposals
L08 Collaboration
•public
•inclusive
L07 Maintenance
•releases
•packaging
•release notes
L06 Entry Barriers & Communication
•recognice
•improve
L05 Documentation
•generation
•verification
L02 Source Code Management
merge
pull
push
pull
L03 Build Tools
+shell scripts
+reformatting
> .....
.....
> .....
> cd /
L04 Continuous Integration
building
coding
static checks
fuzzer
testing
•code coverage
•test automation
•pull requests
•decentralized
•Bug triage
•Bug reports
L01 Issue Tracking

Overviewability

- complementary to traceability
  (i.e. trace from requirements/documentation/issues to affected code and back)
- measurement how long a newcomer needs to find her/his way
- answers: “Where to add new functionality?”
- in FLOSS directory structure in repository essential (= physical view)

Grouping of Files

- by programming language
- by topics (tests, src, doc)
- by modules/plugins

Metadata of Files and Directories

- README.md
- file name endings
- LICENSES and `.license`
- `scripts` vs. `src` (executable bits)

Cleanup

- top-level
- main folders
- file names
- automatic formatting (encodings and line endings)
- reduce dependencies between folders

But also stop worrying if needed by convention.

Logical Views

- Documentation Generators (“Files” in Doxygen is physical view)
- Building Block View (may be identical to physical)
- Runtime View
- Deployment View

L09 Architecture

Markus Raab

Institute of Information Systems Engineering, TU Wien

This work is licensed under a Creative Commons *"Attribution-ShareAlike 4.0 International"* license.

Software Architecture

1. Directory Structure
2. Software Architecture
3. Architectural Decisions
4. Meeting
   - Recapitulation
   - Assignments
   - Preview

Software Architecture

- architecture is a high-level description of the overall system
- use ready-made patterns and templates for architecture
- e.g., `http://arc42.org/`

Arc42

1. Introduction and Goals
2. Constraints
3. Context and Scope
4. Solution Strategy
5. Building Block View
6. Runtime View
7. Deployment View
8. Crosscutting Concepts
9. Architectural Decisions [1]
10. Quality Requirements
11. Risks and Technical Debt
12. Glossary

Example

Crosscutting concept “configuration settings”:

- are stored in configuration files
- in data structure KeySet
- modified by configuration management tool using KeySet

more about it in course “configuration management”

Roles

In FLOSS usually nobody is project manager
$\rightarrow$ but everyone is software architect

Goals

The most important tasks of software architects are

- to pursue the right goals
- to have good documentation (e.g. with arc42)
- to keep everything as simple as possible
- to communicate the architecture
- maintain community and quality

Refactoring

- build what community needs at the moment
- change according to current needs
- avoid over-engineering, refactor to KISS

L09 Architecture

Markus Raab

Institute of Information Systems Engineering, TU Wien

This work is licensed under a Creative Commons *"Attribution-ShareAlike 4.0 International"* license.

Architectural Decisions

1. Directory Structure
2. Software Architecture
3. Architectural Decisions
4. Meeting
   - Recapitulation
   - Assignments
   - Preview

Architectural Decisions

- describe decisions that lead to the architecture
- decisions are high-level configuration
- patterns/templates are useful [1], e.g.:

Template

1. problem
2. constraints
3. assumptions
4. considered alternatives
5. decision
6. rationale
7. implications
8. related
9. notes

Example: API Design

- future-proof
- hard to use it wrong vs. easy to use
- consistent concepts, e.g. for resources
- minimal vs. comfort

Example: Libraries vs. Daemons

- both foster reuse of code
- daemon better if there is dynamic state
- but: daemon creates a single point of failure (KISS)

Dangers

> *Insanity in individuals is something rare – but in groups, parties, nations, and epochs, it is the rule. – Friedrich Nietzsche*

In groups you get confronted with the whole spectrum of psychology:

- Groupthink (conformity)
- Group polarization

L09 Architecture

Markus Raab

Institute of Information Systems Engineering, TU Wien

This work is licensed under a Creative Commons *"Attribution-ShareAlike 4.0 International"* license.

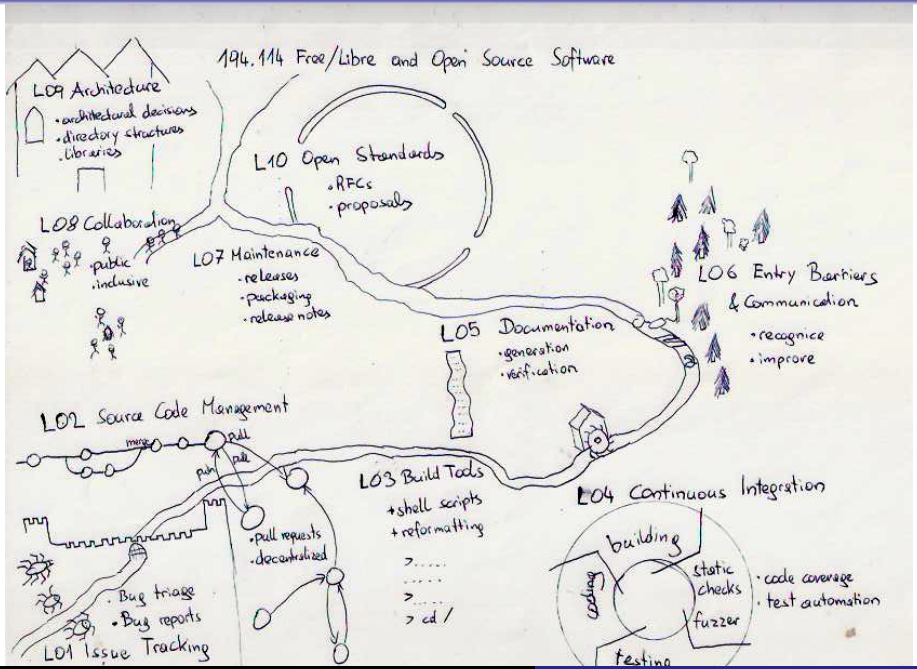
Meeting

1 Directory Structure

2 Software Architecture

3 Architectural Decisions

4 Meeting
- Recapitulation
- Assignments
- Preview


194.114 Free/Libre and Open Source Software
L09 Architecture
•architectural decisions
•directory structures
•libraries
L10 Open Standards
•RFCs
•proposals
L08 Collaboration
•public
•inclusive
L07 Maintenance
•releases
•packaging
•release notes
L06 Entry Barriers & Communication
•recognice
•improve
L05 Documentation
•generation
•verification
L02 Source Code Management
merge
pull
push
pull
L03 Build Tools
+shell scripts
+reformatting
> .....
.....
> .....
> cd /
L04 Continuous Integration
building
testing
static checks
fuzzer
•code coverage
•test automation
•pull requests
•decentralized
•Bug triage
•Bug reports
L01 Issue Tracking

Learning Outcomes

After successful completion of L09 Architecture students will be able to

- reproduce chosen FLOSS software architectures.

Overviewability

What is traceability and what is overviewability?

Overviewability

- complementary to traceability
  (i.e. trace from requirements/documentation/issues to affected code and back)
- measurement how long a newcomer needs to find her/his way
- answers: “Where to add new functionality?”
- in FLOSS directory structure in repository essential (= physical view)

Discussion: How to Group Files?

- by programming language/projects (monorepo)
- by topics (tests, src, doc)
- by modules/plugins

Task

How do you like files to be structured?
How do you keep order over time?

Reading Text

Task

Summarize what you found interesting.

Task

Break.

Arc42

1. Introduction and Goals
2. Constraints
3. Context and Scope
4. Solution Strategy
5. Building Block View
6. Runtime View
7. Deployment View
8. Crosscutting Concepts
9. Architectural Decisions [1]
10. Quality Requirements
11. Risks and Technical Debt
12. Glossary

Goals

Task

What are the most important tasks of software architects?

Goals

The most important tasks of software architects are

- to pursue the right goals
- to have good documentation (e.g. with arc42)
- to keep everything as simple as possible
- to communicate the architecture
- maintain community and quality

Architectural Decisions

Task

What are architectural decisions?
Please given an example.

Architectural Decisions

- describe decisions that lead to the architecture
- decisions are high-level configuration
- patterns/templates are useful [1], e.g.:

Task

Break.

T3 H4 P3

Task

Get all your PRs ready to merge.

Feedback

Today home and teamwork will finish for this term.

- Feedback Talk
- ECTS breakdown realistic?

L10 Open Standards

[1] Neil B Harrison, Paris Avgeriou, and Uwe Zdun. Using patterns to capture architectural decisions. *Software, IEEE*, 24(4):38–45, 2007. ISSN 0740-7459. doi: 10.1109/MS.2007.124.