Advice on Preparing Reports in Softwareengineering

Paul Fischer, January 2023

This document is in English, as it is also used in master-courses.

General

Avoid to only give a manual for or description of the final product. Instead explain why the product looks as it is. Document the development process, decisions made on the way, and how you tackled problems.

The points below do not have to (but might) be the headlines/sections of the report and some might not be relevant for some projects. Below, the word "describe" has to be understood as "describe and justify your choice".

Remember that the person who is responsible for the course has to look at all reports, use appendices for lengthy illustrations. **Avoid to use internal "slang" expression, which you have developed during the project, without clearly defining them for the reader.**

Document, who is responsible for which part. It is **not** enough to say something like "we all contributed to the same extend". Best to set names at the start of sections/paragrahps.

Make sure that your names, study numbers, and group number appears on the cover page.

The problem

- Describe the problem and the context/domain in which it appears.
- Describe the scope of the project: Which sub-problems will be considered, which will not be considered.

The goals

- (Functional) What are the main goals of the project, what should be the properties of the final product?
- (Non-functional) Usability, performance, maintainability, extensibility, simplicity, security, ...
- (Non-project) Learn new technology, learn project management, ...
- What are the success criteria for the project?
- What is your level of ambition, both with respect to the problem and the grade?

Analysis of the problem

This is independent of the type of project (hard-/software) and the HW-platform or programming language (unless enforced by external factors).

- What are the main challenges for solving the problem.
- What are potential sub-problems.
- Are there know solutions to sub-problems.
- Propose solutions to the problem/sub-problems, anme alternatives.
- Select the solutions you want to use and **justify you choice**.

Its is important that you explain why you selected or discarded a solution.

Design

The design should be largely independent of the implementation environment/paramming language. How ever the design paradigm (for example Model-View-Controler) has impact here.

- Describe the general architecture of the solution.
- Identify major system components (components, packages, interfaces, ...)
- Describe the technologies/platforms used.

This serves as a guideline for the implementation.
Again, discuss alternatives and explain why you selected or discarded them.

Implementation

This turns the design into software (or hardware).

- Describe the system components.
- Specific algorithms and data structures.
- If applicable, describe detailed class hierarchies or data base schemes.
- Avoid to describe the whole code in detail (this can be done in an appendix and as comments in the code listing), however point out unconventional coding.

Test

- Which types of test were used and why were they chosen?
- Document the results and potential changes because of errors found.

Project Management

- Did you use project management paradigms/tools, if not why?
- How did you assign tasks to the group members and how was the quality of the devirances checked?
- How did it go, if there were conflicts, how were they handled?
- How did you manage time?

Conclusion

- What was achieved, which goals were reached.
- What was not achieved and why.
- What could have been done better ("if we could start again then . . . ").
- Possible extension, ideas for the future.
- What did you learn in the project, both professionally and socially?
- Individual, personal experiences, "I . . . . . . ".