Software Construction Techniques
Writing Good Code

Paolo G. Giarrusso
Prof. Klaus Ostermann
In this seminar, you will learn about
• software construction
• giving presentations
• writing short papers on the topic
• writing reviews on other papers
A core problem of writing software, in the abstract

• Software projects can be too complex
• Successful software projects must control that complexity
• Otherwise, programs will be too complex to modify successfully
Good Code & Programmers

• Many programs are equivalent…
• … but not for humans…
• What’s the difference?
• How can you write code that others enjoy reading, rather than suffer through?
Prototypical Example: Code Duplication

- If you need the same code twice, you can
  - copy-n-paste
  - abstract the code into reusable form (e.g. a routine)
- Both “work”
- Copy-n-paste will cause lots of pain down the road
  - Why?
    - More effort during maintenance
    - More effort during understanding
- Is abstraction always worthwhile?
Coding Religions

• *Gurus and zealots*...
• ... *evangelize* you to follow *mantras*...
• ... and promise *salvation* (aka *silver bullets*).

In fact
• few absolute rules
• tradeoffs to understand
Tradeoff Example: Automation

• Automate tasks that can be automated to save developers’ time

• But…
"I spend a lot of time on this task. I should write a program automating it!"

**Theory:**
- Writing code
- Work on original task
- Automation takes over
- Free time

**Reality:**
- Writing code
- Debugging
- Rethinking
- No time for original task anymore

https://xkcd.com/1319/
Tradeoff Example: Automation

• Possible Solution: Automate tasks that need be done **consistently**
  • Building software
  • New releases
  • Testing…
Tradeoffs, Therefore...

• Learn to debate the reasons of practices
Software Construction Versus Software Engineering

• *Software construction* is one task in creating software (*software engineering*)
  • The down-to-Earth part
  • Often neglected
  • The part that happens in every project

• So we will not focus on development process, requirements
Sources

*The Pragmatic Programmer*

- Lots of programming wisdom
- Guru-like, but mostly right
- Enjoyable read
- Short & a bit chaotic
Sources

*Code Complete*

- Complete & systematic
- Tries hard to be evidence-based
Sources: Further material

Caveat:

• Both are OOP-based
• These books don’t necessarily assume a CS education
Some (Possible) Topics

• High-Quality Routines (CC Ch. 6)
  • Cohesion, naming, size, parameters…

• Organizing Statements (CC Part IV)
  • Including the debate on goto

• Self-documenting code (CC Ch. 32)

• Programming Character (CC Ch. 33)

• Code reviews (CC Ch. 21)
Seminar Format
Seminar Goals

• Learning about the topic
  • Here, about programming

• Learning how to do scientific work
  • Here, reflect upon programming advice, don’t trust it blindly
Format

Scientific work consists of:

- Read & understand
- Think & create
- Write & reflect
- Discuss & convey
Seminar...

• Read & understand: ✓
• Think & create: ✗
• Write & reflect: ✓
• Discuss & convey: ✓
...vs thesis work

- Read & understand: ✓
- Think & create: ✓
- Write & reflect: ✓
- Discuss & convey: mostly ✗

Yet, a seminar can be useful preparation for thesis work.
Tentative Schedule

1. Kick-off
2. (Topic Choice?)
3. Preparation on Writing
4. Preparation on Presentations
5. … your presentations
Grading

- 40% talk
- 40% paper
- 20% reviews & participation