Software Engineering

Lecture 01 – Introduction

© 2015-20 Dr. Florian Echtler
Bauhaus-Universität Weimar
<florian.echtler@uni-weimar.de>

Why software engineering?

Image source (CC): https://www.flickr.com/photos/jacobavanzato/16152519186

- Software development is (relatively) easy
- (Unlike operating a chainsaw in the forest)

Why software engineering?

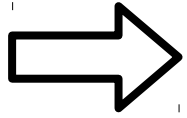
Image source (CC): https://www.flickr.com/photos/78044378@N00/364003706

- A first prototype is usually finished quickly …

Why software engineering?

Image sources (CC): https://www.flickr.com/.../7165270428, https://commons.wikimedia.org/...Wohnhaus_01.jpg

- … but expanding that prototype is often much more difficult.



Why software engineering?

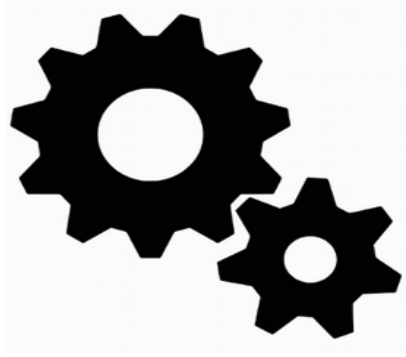
- Create software that is …
  - well-structured
  - easy to maintain
  - (re-)usable
  - … and meets its requirements.
- Provide a development process which is …
  - predictable
  - adaptable
  - … and on time.

Topics (1)

- Object-Oriented Programming
- Best practices
  - revision control: SVN, git, …
  - testing: unit tests, iterative development, …
  - UML diagrams (refresh from Modelling class)
- Development paradigms/process models
  - “Classic”: Waterfall, RUP, Spiral, …
  - “Agile”: Scrum, XP, …

Topics (2)

- Design Patterns (in Java)
  - simple patterns: e.g. *Singleton, Factory*
  - complex patterns: e.g. *Visitor, Observer*
- Miscellaneous Topics
  - Building & Debugging
  - Requirements Engineering
  - UI Patterns
  - Open-Source software

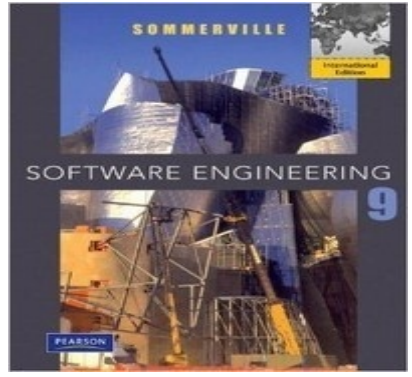


Lectures

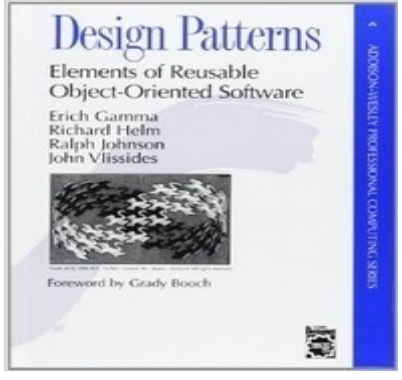
- Introduction
- Object-Oriented Development
- Best practices: RCS, testing, UML
- “Classic” & “agile” development models (2 lectures)
- Design Patterns (2 lectures)
- Code Quality & Debugging
- Build Systems & Continuous Integration
- Testing & Requirements Engineering
- Open Source Software

Companion Books

- Ian Sommerville, “Software Engineering” (9th edition)
- http://opac.ub.uni-weimar.de/DB=1/PPNSET?PPN=618878599



- Erich Gamma et al., “Design Patterns”
- http://opac.ub.uni-weimar.de/DB=1/PPNSET?PPN=520726987



Further reading

- Fred P. Brooks, “The Mythical Man-Month” [1]
  (historic perspective on SE)
- Eric S. Raymond, “The Cathedral and the Bazaar” [2]
  (open source development)
- Kent Beck et al., “The Agile Manifesto” [3]
  (origins of agile development)

[1] http://opac.ub.uni-weimar.de/DB=1/PPNSET?PPN=352067705

[2] http://www.catb.org/esr/writings/cathedral-bazaar/

[3] http://agilemanifesto.org/

Famous software failures

- Software is *everywhere*

  → *massive* damage potential

- Billions of € lost
- Hundreds of deaths & injures

Ariane 5 Flight 501

Image source (FU): https://www.ima.umn.edu/~arnold/disasters/ariane.html

- Left/center: liftoff (June 4th, 1996)
- Right: self-destruct after 39 seconds
- ~ 500 million US-$ damage

Ariane 5 Flight 501

- Reasons: integer overflow
  - 64-bit float velocity converted to 16-bit integer
  - Measurement > 32768 → overflow →
    → uncaught exception → system crash
- Software reused from Ariane 4
  - Old rocket was slower, overflow impossible
  - No overflow handler → system crash with A5

Therac-25

Image source (FU): http://cr4.globalspec.com/blogentry/19025/Failure-of-the-Therac-25-Medical-Linear-Accelerator

- Radiation therapy machine for cancer patients
- At least 6 accidents with massive radiation overdoses
  → death or serious injury

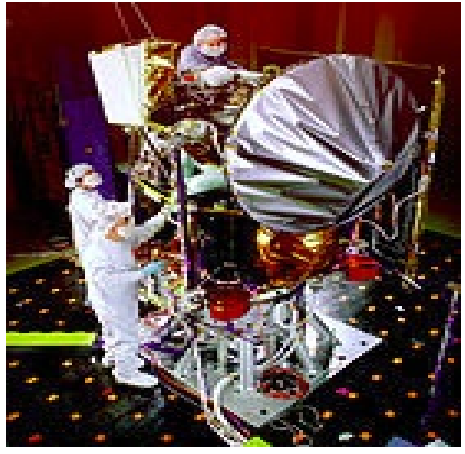
Therac-25

- Critical safety checks in software only
- 1 single developer, no formal tests
- Race condition:
  - Operator enters wrong radiation intensity/mode
  - Machine starts to setup radiation beam
  - Operator corrects error within ~ 8 seconds
  - Display shows new value, but setup unchanged

Mars Climate Orbiter

Image source (PD): https://en.wikipedia.org/wiki/Mars_Climate_Orbiter

- Lost during Mars approach 1999
- ~ 330 million US-$ damage



Mars Climate Orbiter

- Reason: “Because of the metric system!”
- Ground control software:
  - Module A: expects data in metric SI units
  - Module B: sends data in Imperial units

→ trajectory calculations wrong

→ probe burns up in Mars atmosphere

Schiaparelli Lander

Image source (CC): https://commons.wikimedia.org/wiki/File:Schiaparelli_Lander_Model_at_ESOC.JPG

- Lost during Mars landing 2016
- ~ 100 million US-$ damage

Schiaparelli Lander

- Fast spin after parachute deployment
- Overflow of onboard spin sensor

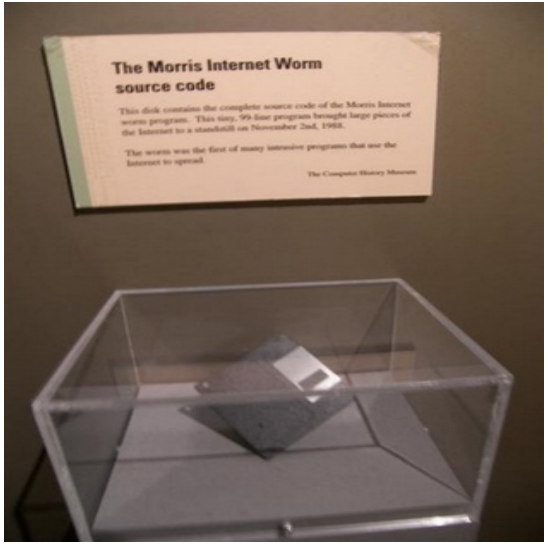
  → negative height calculation
- Parachute separation at ~ 1 km

  → Crash

“Morris Worm”

Image source (CC): https://en.wikipedia.org/wiki/Morris_worm#/media/File:Morris_Worm.jpg

- First “computer virus” in 1998 - by accident
- Research tool to estimate Internet size



“Morris Worm”

- Heuristic: in 1 of 7 cases, install a second copy of the worm (as anti-removal measure)
- Far too high: most infected systems had dozens of copies running
- Massive slowdown of the entire 1998 Internet

Patriot Missile Disaster

Image source (CC): https://en.wikipedia.org/wiki/MIM-104_Patriot

- Error in guidance system
- Resulted in 28 deaths during 1991 Gulf war

Patriot Missile Disaster

- Internal timestamps from radar detection
- Only 24-bit precision floats
- Conversion errors accumulate over time
- Final error after ~ 100 h: 0.3 seconds
- 0.3 s = ~ 600 m missile travel distance

AT&T phone network crash

Image source (FU): http://www.att.com/gen/press-room?pid=6158&cat=46&u=484

- AT&T network collapses on Jan 15th, 1990
- 11 h downtime, ~ 60 million US-$ damage

AT&T phone network crash

- Software update on phone switches
  - 1. Switch detects an error
  - 2. Switch sends “congestion signal” to peers
  - 3. Switch resets itself
  - 4. Switch starts to forward calls again
- Problem: peer switches crash themselves when receiving multiple messages in 4.
- Cascading failure across whole network

Boeing 737 MAX

Image source (CC): https://de.wikipedia.org/wiki/Datei:Boeing_737-8_MAX_N8704Q_rotated.jpg

- Uses “Maneuvering Characteristics Augmentation System” (MCAS)
- Software “patch” for dangerous flight behaviour due to newer engines

Boeing 737 MAX

- Two “angle-of-attack” sensors, but only one used by MCAS (?)

  → faulty sensor value → unnecessary activation
  → (confused pilots) → crash

Toyota Acceleration Bug

Image source (FU): http://www.toyota-global.com/showroom/emblem/passion/

- Causes “unintended acceleration”
- May have caused up to 89 deaths

Toyota Acceleration Bug

- Internal software essentially untestable
  - 10 000 global variables
  - No hardware bit flip protection
  - Single process for multiple tasks
  - Improper process management
  - Multiple Single-Points-of-Failure

Summary

- Software is everywhere
- Software ***must*** be reliable
- Many errors are obvious in hindsight
- How can we prevent them?

Questions/suggestions?