Welcome to
Software Engineering

Thomas Fritz

Reid’s reading “Course intro”:

Many thanks to Reid C. Holmes and Elisa Baniassad (both UBC)
Agenda

1. Why Software Engineering?
2. What is Software Engineering?
3. Course Overview
Why Software Engineering?

The Ariane 5 ....

http://youtu.be/gp_D8r-2hwk
http://www.di.unito.it/~damiani/ariane5rep.html
An Uncaught Exception!?

Sadly, the primary cause was found to be a **piece of software which had been retained from the previous launchers systems and which was not required during the flight of Ariane 5.**

The software was used in the Inertial Reference System (SRI) to calculate the attitude of the launcher. In Ariane 4, this software was allowed to continue functioning during the first 50 seconds of flight as it could otherwise delay launching if the countdown was halted for any other reason, this was not necessary for Ariane 5. As well, the software contained implicit assumptions about the parameters, in particular the horizontal velocity that were safe for Ariane 4 but not Ariane 5.

The failure occurred because the horizontal velocity exceeded the maximum value for a 16 bit unsigned integer when it was converted from it's signed 64 bit representation. This failure **generated an exception in the code which was not caught and thus propagated up through the processor and ultimately caused the SRI to fail.** The failure triggered the automatic fail-over to the backup SRI which had already failed for the same reason. This combined failure was then communicated to the main computer responsible for controlling the jets of the rocket, however, this information was misinterpreted as valid commands. As a result of the invalid commands, the engine nozzles were swung to an extreme position and the launcher was destroyed shortly afterwards.

**The failure was thus entirely due to a single line of code.**
“Self-Driving Tesla Was Involved in Fatal Crash, U.S. Says”


“the crash occurred when a tractor-trailer made a left turn in front of the Tesla, and the car failed to apply the brakes”

“Software is eating the world” (Marc Andreesen)
Software is everywhere
What is Software Engineering?

The process of transforming a mental plan of desired actions for a computer into a representation that can be understood by the computer.

— Jean-Michel Hoc and Anh Nguyen-Xuan
Software is growing
Codebases
Millions of lines of code

hundred thousand

- simple iPhone game app
- Unix v 1.0 1971
- Win32/Simile virus
- average iPhone app
- Pacemaker
- Photoshop v. 1.0 1990
- Camino web browser
- Quake 3 engine 3D Video game system
- Space Shuttle

from http://www.informationisbeautiful.net/visualizations/million-lines-of-code/
1 million lines of code
18,000 pages of printed text

1. War And Peace x 14, or Ulysses x 25, or The Catcher in The Rye x 63

2. CryEngine 2
   3D video game system

3. Bacteria
   Syphilis (Treponema pallidum)

4. Age of Empires online

5. CESM Climate Model
   National Center for Atmospheric Research

6. F-22 Raptor fighter jet

7. Linux Kernel 2.2.0
   core code

8. Jurassic Park codebase
   source: Dennis Nedry

9. Hubble Space Telescope

10. Unreal engine 3
    3D video game system

11. Windows 3.1
    1992

12. Large Hadron Collider
    (root software)
from http://www.informationisbeautiful.net/visualizations/million-lines-of-code/
What is Software Engineering?

People working **together**, to **create** a **robust** software system that **satisfies the client**.

This involves **technical** and **interpersonal** challenges!
Small class activity

Have you ever developed software for a company?

- If so, what SE practices did they / you use? (e.g. source control, coding standards, processes, ...)

Why are these practices useful?
Course Overview
About me (Thomas)

Associate Professor (since January 2018)
- (previously) Assistant Prof at UZH and University of British Columbia
- PhD at UBC in 2011 (originally from Munich)

Research area: software engineering, particularly
- Developer productivity & retrospection
- Biometric sensing
- Information needs

Research with multiple companies: ABB, Microsoft, IBM, Biovotion, Intearxon, ...

Room: 2.B.21, Email: fritz@ifi.uzh.ch

Office hours are by appointment. Email me or get in touch with me after class.
Course Assistants and TAs

- Manuela Züger (zueger@ifi.uzh.ch)
  - PhD student
- Lukas Eisenring
- Christoph Vogel
### Tentative Course Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.02</td>
<td>Introduction to Software Engineering</td>
</tr>
<tr>
<td>28.02</td>
<td>Process &amp; Agile</td>
</tr>
<tr>
<td>07.03</td>
<td>Specification (User Stories)</td>
</tr>
<tr>
<td>14.03</td>
<td>Decomposing User Stories</td>
</tr>
<tr>
<td>21.03</td>
<td>Modularity &amp; High Level Design</td>
</tr>
<tr>
<td>28.03</td>
<td>REST APIs &amp; Detailed Design</td>
</tr>
<tr>
<td>04.04</td>
<td><strong>Easter holidays</strong></td>
</tr>
<tr>
<td>11.04</td>
<td><strong>Midterm</strong></td>
</tr>
<tr>
<td>18.04</td>
<td>Design Principles, SOLID</td>
</tr>
<tr>
<td>25.04</td>
<td>Design Patterns</td>
</tr>
<tr>
<td>02.05</td>
<td>Refactoring &amp; Smells</td>
</tr>
<tr>
<td>09.05</td>
<td>Testing and Software Quality</td>
</tr>
<tr>
<td>16.05</td>
<td>Software Evolution</td>
</tr>
<tr>
<td>23.05</td>
<td>Cloud Deployment &amp; Release</td>
</tr>
<tr>
<td>30.05</td>
<td>Recap / Advanced Topics</td>
</tr>
</tbody>
</table>
This course is about...

- Software engineering, software processes, steps in software development, **design and modularity**, testing and more

- By the end you should be able to
  - Demonstrate knowledge and explain challenges of the steps involved in building a large, complex software system
  - Understand benefits and disadvantages of various designs for constructing software systems
  - Communicate technical matters with programmers, managers, and clients effectively; and convey the rationale behind your choices

- Taking software construction to the next level (team, size, ...)

Wednesday sessions – somewhat *flipped* classroom

Each week before class

- Watch videos / read readings in the week before each lecture
- Videos on edX, readings on web page
- Most material from Reid Holmes (colleague at UBC)
- Fill in survey by **midnight Sunday** (part of participation grade)

  Please list any questions that you have regarding the pre-reading / video assignment, the pre-class problems, or how this week's content relates to the rest of the course material. You must submit at least one question.

In case there is no reading/video ➔ no survey
Wednesday sessions – somewhat flipped classroom

Each week in class (bring your laptop)

- Recap
- Predominantly activity based
- Will be great practice for midterm and final

- less listening, more practice
- this is different to before and we are trying it out, so please bare with us
Prior Knowledge & More

Students should have completed software construction

You should be familiar with Object-Oriented Programming

This course will use Java for examples
Grading

25% Midterm (11.4, 10am – 12pm)
65% Final (20.6, 10am – 12pm)

midterm & final most likely multiple choice; have no fear, since multiple choice allows us to ask more questions, which makes each question worth less and lets you excel at parts where you’re confident; it also means faster feedback and less arbitrary marking.

10% Participation (weekly survey + 2 assignments)

each survey has one question; will be evaluated for completeness; to get credit, we will ask: “could this answer have been given for any other week?”
Resources


OLAT [https://lms.uzh.ch/auth/RepositoryEntry/16371188499](https://lms.uzh.ch/auth/RepositoryEntry/16371188499)
- Links to videos / readings
- Survey & assignments
- Forum (do NOT post solutions and do NOT use it for personal issues that need to be discussed with me)
- Other material

In class resources will **NOT** be distributed
Reading on course / videos will take the place of slides
SE and SoPra

- First year we put this into one term

- Not yet in sync + not everyone takes both
  - Please do not expect it, some material will be covered here in more detail, only after it was part of SoPra

- We are working on syncing it and I will try to put the reading / video material up beforehand
Things to do THIS WEEK

- Read the introduction to software engineering reading
- *Optional: listen to programming language introduction video*

- Listen to the Process module videos (create edX account for this); additionally provided reading also captures most of the video material
  - Why Process?, Traditional Processes (Waterfall & Spiral), Agile Processes (XP and TDD), Scrum → 6 videos (1 to 6mins long)

- Fill out first survey by Sunday (25.02) midnight