# Software Engineering

## Lecture 01: Introduction

Peter Thiemann

University of Freiburg, Germany

SS 2014

# Introduction

## Software Engineering

- Programming in the large (DeRemer, Kron, 1975)
- Principles, models, and techniques for development and maintenance
- Emphasis on engineering techniques
- Goals:
  - meeting the requirements (functionality, quality, efficiency, etc)
  - delivering on time
  - reducing the cost of development and maintenance (often 80 %)

## Software Crisis

- Coined 1965, NATO meetings 1968/69
- Programs hard to maintain
- Documentation absent or obsolete
- Overrunning cost and deadlines

# Characteristics of Software

**"Software is soft"**

- Immaterial
  - no wear and tear
  - no physical limitations
  - hard to measure
- Changeability: easy or hard?
- Aging

# Software Development Today

- Hardware not an issue in the days of Moore's law
  - Cost for development and maintenance more important than efficiency
    cost of software $\nearrow$, cost of hardware $\searrow$
  - More complex systems
- Teamwork essential ($\rightarrow$ decomposition, interfaces, contracts)
- Diverging classes of applications

# Two Extreme Types of Applications

## Applications with short time-to-market

Base functionality more important than correctness or robustness
$\rightarrow$ accelerated "agile" design process

## Safety critical applications

Correctness essential for functionality
$\rightarrow$ (semi-) formal methods, verification

# Software Crisis Today . . .

# Bugs can stop you: USS Yorktown (CG-48)

In September 1997, a crew member of the cruiser USS Yorktown mistakenly entered a zero for a data value, which resulted in a division by zero. The error cascaded, crashed every computer (WinNT 4.0), and eventually shut down the ship's propulsion system. The ship was dead in the water for 2 hours 45 minutes.

# Context

## Smart Ship Project (1996)

- 27 terminals w/ Dual Pentium Pro 200MHz processors running WinNT 4.0
- Connected by fibre-optical network
- Ship can be controlled from every terminal
- Purpose of system: supervision of ship including propulsion

# Context

## Smart Ship Project (1996)

- 27 terminals w/ Dual Pentium Pro 200MHz processors running WinNT 4.0
- Connected by fibre-optical network
- Ship can be controlled from every terminal
- Purpose of system: supervision of ship including propulsion

## Motivation

- Cut down personnel (10% out of 400)
- Cut down cost (by 2.8 Mio USD)

# How did it happen?

- Software reports “gauge open”, but the gauge is closed

# How did it happen?

- Software reports “gauge open”, but the gauge is closed
- To fix: officer performs **direct modifications** of the ship’s database

# How did it happen?

- Software reports “gauge open”, but the gauge is closed
- To fix: officer performs **direct modifications** of the ship’s database
- A particular entry was changed to “0”

# How did it happen?

- Software reports “gauge open”, but the gauge is closed
- To fix: officer performs **direct modifications** of the ship’s database
- A particular entry was changed to “0”
- Caused a division by zero elsewhere in the system

# How did it happen?

- Software reports “gauge open”, but the gauge is closed
- To fix: officer performs **direct modifications** of the ship’s database
- A particular entry was changed to “0”
- Caused a division by zero elsewhere in the system
- Caused buffer overflow and overwrote propulsion data

# How did it happen?

- Software reports “gauge open”, but the gauge is closed
- To fix: officer performs **direct modifications** of the ship’s database
- A particular entry was changed to “0”
- Caused a division by zero elsewhere in the system
- Caused buffer overflow and overwrote propulsion data
- Bingo!

# Bugs can be fatal

In September 1993 an A320 skidded off the end of the runway during landing. The aircraft touched down with sink rate low enough that the onboard flight computers did not consider it to be "landing", which inhibited thrust reverse and brake application for nine seconds.

The failure was caused by two preconditions for braking (weight on both wheels, wheels spinning). They were introduced because of an earlier accident (Lauda Air) where reverse thrust was applied during normal flight, wrongly assuming the plane was about to land.

# Bugs can be expensive: The Ariane 5 Failure on June 4, 1996

- The first launch of the Ariane 5 rocket was an expensive failure
- Video: `https://www.youtube.com/watch?v=gp_D8r-2hwk`
- Video explanation: `https://www.youtube.com/watch?v=W3YJeoYgozw`

# Sometimes a bug is more than a nuisance

James Gleick `http://www.around.com/ariane.html`

It took the European Space Agency 10 years and $7 billion to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business. All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a small computer program trying to stuff a 64-bit number into a 16-bit space. [. . . ] At 39 seconds after launch, as the rocket reached an altitude of two and a half miles, a self-destruct mechanism finished off Ariane 5, along with its payload of four expensive and uninsured scientific satellites. Self-destruction was triggered automatically because aerodynamic forces were ripping the boosters from the rocket.

# Conclusion?

## Recurring themes

- Unclear requirements
- Buggy specification
- Oversights in testing
- Unforeseen (human) interaction with the system
- Not in the examples: time pressure, stakeholder pressure, financial pressure

# Conclusion?

## Recurring themes

- Unclear requirements
- Buggy specification
- Oversights in testing
- Unforeseen (human) interaction with the system
- Not in the examples: time pressure, stakeholder pressure, financial pressure

## Software Engineering

- Supposed to provide cure for all that
- And more ...
- Mission Impossible?

# State of Affairs

- No consensus on "what is software engineering"
- No simple (or single) answer
  ("No silver bullet", Fred Brooks, 1987)
  `http://www.cs.nott.ac.uk/~cah/G51ISS/Documents/NoSilverBullet.html`
- But there are **techniques**, **methods**, and **tools**, that can reduce the complexity of constructing systems
- There are also techniques for building specific kinds of systems with high degrees of reliability

# Approach

- It is not possible to present and practice the full spectrum of approaches to software engineering in one class
  - industrial setting is completely different from a university
  - insufficient time for development in the large
  - insufficient experience with software development
  - different problems demand different techniques

$\Rightarrow$ We survey central concepts and experiment with selected approaches

$\Rightarrow$ Emphasis on techniques for safety critical systems

- Specialized techniques presented in advanced courses

# Curriculum

1. Introduction
   - Activities in SW development
   - Software development processes
2. Requirements & Specification
   - Use cases, use case diagrams (UML), user stories
   - Overview of the B specification method
   - Design by contract, code contracts, monitoring, verification
   - Types, invariants, (type state, session types)
3. Design
   - UML: Data modeling, behavioral modeling, OCL
   - SW Architecture, patterns
   - MDE basics, meta modeling
4. Construction (Implementation)
   - Code generation for classes and relations
   - Debugging
5. Testing
   - Unit tests, random testing, DART
   - Test generation