Software Engineering

Lecture 01: Introduction

Peter Thiemann

University of Freiburg, Germany

SS 2013

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
  - Cost for development and maintenance more important than efficiency (Moore's law)
    cost of software $\nearrow$, cost of hardware $\searrow$
  - More complex systems
- Teamwork essential ($\rightarrow$ decomposition, interfaces, contracts)
- Diverging classes of applications

Two Classes 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 expensive

Customers of the Postbank using a “Spar-card” to withdraw money in January 2002 at another bank didn’t have to use their PIN to do so, and there was also no charge on their accounts. This failure was introduced by changes made to adopt to the Euro. Fortunately there was only one exploit of this failure.

In the first night of operation of a new system 28 billion dollars were wrongly transferred to other banks. Only 24 billion dollars could be returned, the remaining money was untraceable.

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 keep you on the ground

20 Feb 2008
Chaos returned to Heathrow Airport as thousands of passengers were hit by a **total breakdown of Terminal 4's baggage handling system**. Economy class travellers arriving at the terminal, which mainly deals with long-haul flights, were told they could only take hand baggage with them — meaning they either had to leave most of their luggage behind or miss their flights.
Around 4,000 passengers were affected by the problem, almost all of them on British Airways flights, and hundreds decided to switch airlines or postpone their trips rather than leaving their luggage behind.
The terminal's automatic baggage sorting system, which uses computer-controlled conveyor belts to send luggage to the right aircraft, **broke down because of a software failure** at lunchtime on Tuesday.
[Source: `http://www.telegraph.co.uk/news/uknews/1579253/Heathrow-engulfed-by-baggage-chaos.html`]

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

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

Plan for a software engineering course

- 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
  - 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