A Software Crisis
Denver
International Airport

• Approved for construction in 1989
• First major airport to be built in the United States in over 20 years.
• Three terminals + several runways
• Built on 53 square miles of land (Twice the size of Manhattan Island!)
BAE Contract

- Original assumption: Every company builds its own baggage transport system
- United (70% Denver traffic) was the only to begin planning; contract with BAE
- First fully automated baggage system
- Later, Denver airport extended contract to entire airport – three times original size
The Scope

• 20 miles of track
• 6 miles of conveyor belts
• 56 laser arrays that read bar coded tags
• 400 frequency readers
• 3,100 standard size baggage ‘Telecars’
• 450 6.5 ft by 4 ft oversize cars
• 55 separate computers
The System
The Timeframe

- BAE started work 17 months before scheduled opening October 31, 2003
- In Munich (similar system), engineers had spent two years just testing the system (with 24/7 operation six months before the airport opened)
More Risks

- Most of buildings were already done, so BAE had to accommodate system (sharp turns, narrow corridors...)
- BAE paid little attention to German sister project and devised system from scratch
- Little communication within BAE
Final Blunder

• The decision to broadcast the preliminary test of the “revolutionary” new baggage system on national television
A Disaster

- Carts jammed together
- Damaged luggage everywhere, some bags literally split in half
- Tattered remains of clothing strewn about caused subsequent carts to derail
- Half the luggage that survived the ordeal ended up at the wrong terminal
More Issues

- Carts got stuck in narrow corridors
- Wind blew light baggage from carts
- 5% of the labels were read correctly
- Normal network load was 95%
Complexity: Empty Carts

• Empty carts need to go where they are needed
• Cart has to be at its “cannon” at the right moment
• Lanes have limited length ➔ traffic jam
• All controlled by single central system
Consequences

• Airport opening delayed four times – overall, sixteen months late
• New engineering firm
  • split system in three (one per terminal)
  • implemented manual backup system
• BAE got bankrupt
• Overall damage: 1.3 bln USD
Glass’ Law

Requirement deficiencies are the prime source of project failures.
Chaos Report

• 31% of projects were *aborted* prior to completion

• in small (large) development companies, *only 16% (9%)* of all projects were completed within projected budget and time limits

Survey by the Standish Group, 1994 – 350 companies with >8000 software projects
Survey by PC week, 1995: 365 information systems professionals on success of software development projects.
More Examples

- **Mariner 1 (1962)**
  Rocket crash due to missing dash
- **Eole 1 (1971)**
  72 weather balloons get wrong cmd
- **Nimbus 7 (1978)**
  Satellite misses ozone hole for 6 yrs
- **HMS Sheffield (1982)**
  Exocet rocket id’ed as “friend”
- **Stanislaw Petrow (1983)**
  Russia detects global nuclear attack
- **Therac 25 (1985)**
  Radiation overdose kills six
- **Stock crash (1987)**
  Dow Jones loses 22% in one day
- **Vincennes (1988)**
  Passenger jet mistaken to be F-14
- **Patriot (1991)**
  Misses to shoot down Iraqi Scud
- **Climate Orbiter (1999)**
  Confuses metrics and imperial
- **US Blackout (2003)**
  50 mln affected for 5 days
- **Apple SSL bug (2012)**
  18 months w/o SSL authentication
How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it

How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed
Challenges

- Why does it take so long to get software finished?
- Why are the development costs so high?
- Why can’t we find all errors?
- Why do we spend so much time and effort maintaining existing programs?
- Why is it difficult to measure progress?
Topics

• Requirements Engineering
• Software Specification
• Software Design and Architecture
• Software Quality Assurance
• Software Maintenance and Evolution
• Software Project Management
Your Lecturers

- Andreas Zeller + Team
- Lecture every Tue 10:15 E2.2...
- ...and sometimes Thu 08:30 E2.2
  (see Web page)
Your Tutors

- Vitalii Avdiienko and Konstantin Kuznetsov (course managers)
- Ezekiel Soremekun Olamide
- Christian Degott
- Doc Cuong Nguyen
- Isabelle Rommelfanger
- Emamurho Ugherughe
- Andreas Thieser
Books

SOFTWARE ENGINEERING
A Practitioner's Approach
Sixth Edition
ROGER S. PRESSMAN

FUNDAMENTALS OF
Software Engineering
SECOND EDITION
Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli
Exam

(+ extra exam beginning of April)
Projects

- SW Engineering is best learned by doing (There is no “theory of software engineering”)
- Therefore, projects make up 2/3 of course
Projects
Supervision
Client
Project Details

- Non-trivial piece of software
- Suggested by *client* (mostly CS members)
- Client is *busy* (spends max 15 hrs total)
- Client is *vague* (on purpose)
Deliverables

- Full set of requirements
- User interface design
- Architecture design
- Project plan
- Prototype
Grading

- Need to pass both exam and project to pass
- Project grades based on group performance (with bonus for individuals)
Software Engineering
Core Course · Winter 2015/2016

Software Engineering Chair (Prof. Zeller)
Saarland University - Computer Science
Campus E1 1
66123 Saarbrücken, Germany
E-mail: se2015-contacts@lists.st.cs.uni-saarland.de

Select a page: SE 2015  Lectures  Projects  F.A.Q.  Exams

News

<table>
<thead>
<tr>
<th>Date</th>
<th>News Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>29 Sep 2015</td>
<td>Course page went live.</td>
</tr>
</tbody>
</table>

Dates and Events

[Calendar for October 2015]
Registration

Registration for the course is mandatory. Please follow the [link](#) to register for the course. **Registration will be closed on Sunday, 25.10.2015 at 23:59:59.**

If you are an international student that cannot register in HISPOS, most probably the proof of course completion you will get from us should be enough for your exchange program coordinator. If you are not sure, contact your coordinator.
Summary

Project Success

Challenges

- Why does it take so long to get software finished?
- Why are the development costs so high?
- Why can’t we find all errors?
- Why do we spend so much time and effort maintaining existing programs?
- Why is it difficult to measure progress?