Software Engineering

[sci-toolset.com]

Instructor: Peter Baumann

email: pbaumann@constructor.university

tel: -3178

office: Research 1, room 88

File not found.
Should I fake it? (Y/N)

First: Semester Timing

- Software Engineering lecture 2.5 CP
  - ~~Tue 08:15~~ (campusnet: project) home office
  - Tue 09:45 (campusnet: project) lecture
- Software Engineering project 5 CP
  - Thu 11:15 (campusnet: lecture) helpdesk session, virtual

June 4, 1996

- Launch of Ariane 5: 4 satellites payload
- 36.7s: Inertial guidance computer crashed
  - Tried conversion 64 bit float → 16 bit integer
  - …but number > 2^15 → overflow
  - Secondary unit: same code → “oops”
- Correcting non-existing deviation → aerodynamic overstress
  - 39s: Emergency self destruction
- Code for pre-flight, shut off after 40secs
- rocket + payload = 300+ mEUR

Prominent Failures and Catastrophes

http://catless.ncl.ac.uk/Risks/ for more

- Y2K
  - 1997 in COBOL stored as „97“; what about ´<´ for 2000?
  - Gartner: worldwide costs $400-$600 billion

Lesson to architect:
don't optimize to death!

- Viking Venus spacecraft: tiny bug in FORTRAN code
  - Correct: DO 20 I = 1,100
  - program code: DO 20 I = 1.100

Lesson to lang designer:
be careful with gimmicks and guessing user intent.
Users need guidance!

- Loss of Mars Orbiter (Sep 1999)
  - incompatible modules - metric vs British / US system

Lesson to developers: *document!*
Lesson to mgrs: *communicate!*

- Instability & security holes of office/Internet sw
  - MS-Word bug: „inability to save doc in only 2% of all cases“

Lesson to management:
quality + cust satisf!

Relevance + Costs

- Economies of ALL developed nations are dependent on software
  - More and more systems are software controlled – lives and economies depend on them…
- Expenditure on sw significant fraction of GNP in all developed countries

- Software costs often dominate computer system costs
- Software costs more to *maintain* than it does to *develop*
  - systems with a long life: maintenance costs may be several times development costs

- *Software engineering concerned with cost-effective software development*

How to Program?
The Code-and-Fix Cycle

1. Write code
2. Test code against a few samples
3. „Improve“ code (bug fixing, extension, efficiency, ...)
4. GOTO 1

- Suitable for 1-person projects and 1st semester
- Maintainability & reliability decrease continuously (“entropy“)
  - Real-life projects: dozens to hundreds of person years
- If developer ≠ user:
  frequent dissent about expected vs. implemented functionality

Common Problems - I

[Jürgen Schönwälder]

- Complexity
  - SAP R/3 (enterprise resource planning sw) as of 1997:
    - *~ 7,000,000 lines of code (Abab, C++),*
    - *~ 100,000 function calls,*
    - *~ 20,000 functions,*
    - *~ 14,000 function blocks,*
    - *~ 17,000 menus*
    - *~ 15,000 database tables*
    - Q: typical deployment time?
  - at main site in Walldorf (Germany) alone several thousand developers
  - size grows by 10% every year
- Integration requirements
  - Ex: B2B / B2C communication; embedded sw
- Quality requirements
  - Ex: Cusumano, MIT Sloan School of Management: 1000 LOC
    - contained ~7-20 defects in 1977;
    - contained ~0.05-0.2 defects in 1994
  - Cf. human transport system: allowed defect rate < 10^{-7}

"If builders built buildings the way programmers write programs, then the first woodpecker that came along would destroy civilization."
~ Weinberg's Second Law

Common Problems - II

[Jürgen Schönwälder]

- Flexibility requirements
  - Change of requirements, components, methods, tools over lifetime
- Portability and internationalization requirements
  - Application software lifetime ~ 10-15 years
    = ~1-2 major changes of underlying system software,
    = ~3-4 changes of underlying hw architecture
- Organizational requirements
  - SW projects usually do not fail because of technical problems
  - Instead, communication problems (customer/developer, developers), lack of education, high fluctuation, or simply bad project management

What is Software Engineering?

- Software Engineering (SE) is
  multi-person construction of multi-version software [Parnas]
- SE as *Engineering* Discipline
  - Adherence to quality standards
  - Role model: classic engineering disciplines
    - *mechanical engineering, civil construction, ...*
- SE offers
  - theory, organisation, management, methods, techniques, tools
  - ...for large program systems

What Software Engineering is NOT

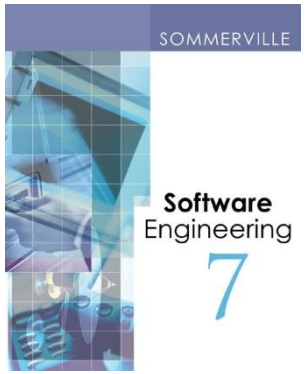
- Not:
  - A continuation of „Algorithms and Data Structures“, smithing cool algorithms
  - Ivory tower thinking
- But:
  - Loads of IT terms
  - Practice that often does not fit a theory
- A culture clash. Sort of.
- Can be painful. In places.

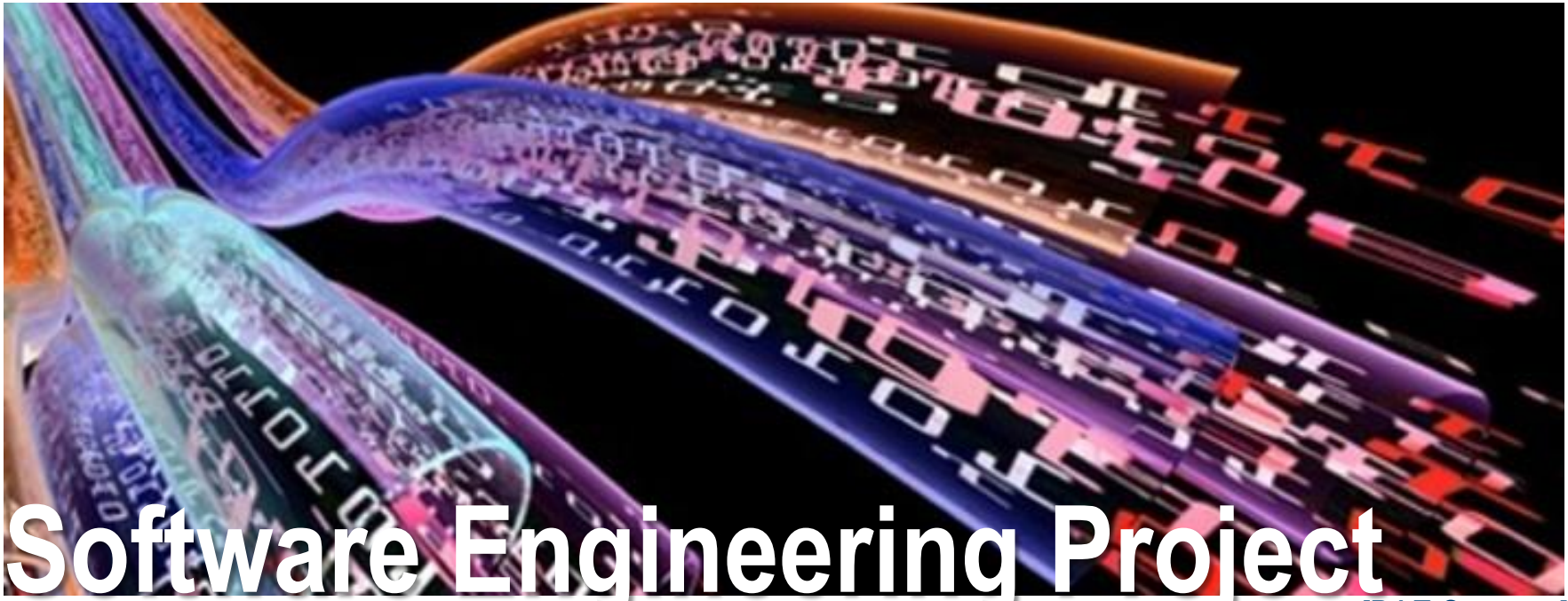
Course Organisation

- Intro
- The software life cycle (plus tool support)
  - Requirements; design (+ patterns); coding; configuration management; verification + testing; maintenance
- Application architectures
  - GUI & Web engineering
- Process & project management
- Wrap-up

References

- Main reference: *Software Engineering*, Ian Sommerville
  - 10th ed., Pearson 2015, ISBN: 978-0137035151
- Some thoughts & slides from:
  Software Engineering – a Practitioner's Approach, Pressman
  - 8th edition, 2014, ISBN: 978-0078022128
- …plus additional material (cited when used)
- Course website: peter-baumann.org
  - Preliminary: https://kahlua.eecs.jacobs-university.de/~pbaumann → teaching
  - Mailing list: course-advcs2@lists.jacobs-university.de
- TAs, instructor





Software Engineering Project

[BAE Systems]

Instructor: Peter Baumann

email: p.baumann@jacobs-university.de

tel: -3178

office: Research 1, room 88

File not found.
Should I fake it? (Y/N)

Project Philosophy

- Goal: as close as possible to industry situation
- Semester project
  - specify + implement + test + integrate + document
- 2-week phases:
  - Specification & design (teams of 5 – free to join up)
  - Code sprints Implementation (teams of 2 – random)
- changing teams, changing code base
  - advancing last-stage code base

„The way to your goal starts the day
you take over 100% responsibility for your actions.“
– Dante Alighieri

How to Start?

- Form groups of 4-5
- Determine spokesperson
- Organise yourself: collaborative editing, meetings, issue list, ...
- When are you done?
  If someone else could build it, without guessing and questions

Project Grading

- Design phase: team submits design document
  - Completeness? Implementability?
  - UML design with use cases, interaction diagrams, class diagrams?
- Coding phase: At each increment
  - automatic extraction from repository
  - Offline code grading
- grades per team
  - *but reserve to deviate in exceptional cases*

Code Grading

- Automatic testing from repository shortly after submission deadline
- Quality checking; see course on quality measures, criteria include:
  - Code compiles, links, runs?
  - Code quality, with criteria such as (!) meaningful class structuring, exception handling, correct output formats, comments, proper formatting, meaningful variable & function naming (1...2-char vars!), ...
  - Amount and quality of documentation
  - Amount & quality of test cases
  - Amount of progress overall, based on (machine-readable) documentation file
  - Plagiarism check

Back to Big Picture

[sci-toolset.com]

"The greatest thing about software is that it is not subject to natural science and that we are all too willing to forget about common sense."

- G. Alonso, VLDB 2003

Module Effort / Grading Scheme

- Software Engineering lecture 2.5 CP exam = 33%
- Software Engineering project 5 CP project = 67%

- Disclaimer:
  Always see CS Handbook of your cohort for authoritative information

So Why Do It?

- Course intends to teach „good craftmanship“ + tools & methods
  - Essential for any IT professional, be it industry or science
  - Coming from a practitioner
- Boost your industrial career perspectives
  - Be sure to have knowledge on hand in job interviews!
  - ...next year I'll add your success story to my teaching page
  - "*consideration of management aspects distinguishes a software engineer over a programmer*" -- Sommerville

"code monkey"

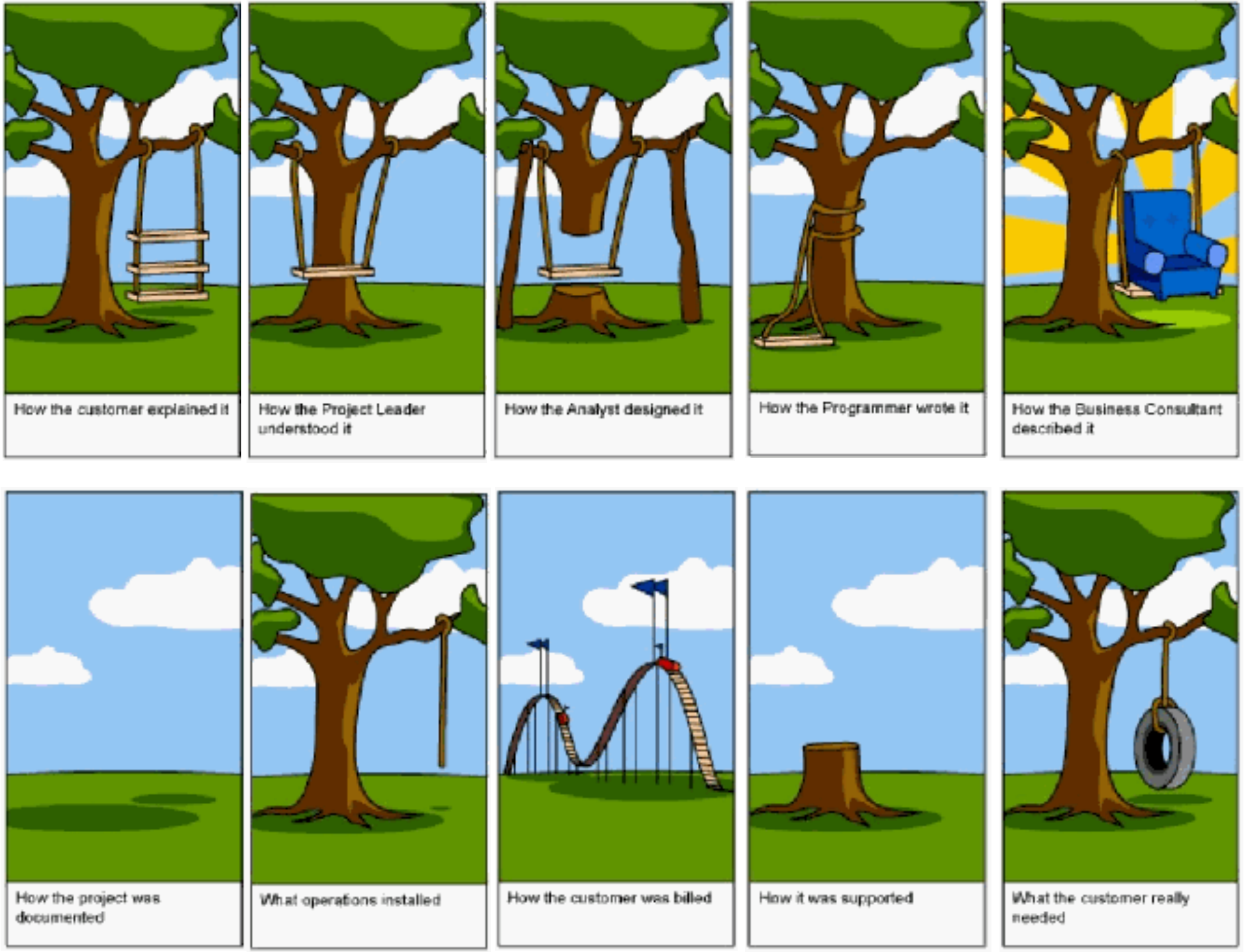
- *PS: Don‘t underestimate!*
  - *Not much math & formulae, but stuff to understand & know & apply*

You'll Make Use of It !

- "I am just in the final stages of my internship at Airbus Hamburg where I developed an online tool to manage the milestones involved in the manufacture of their aircrafts. I was amazed by the fact that I had to use so much of the knowledge I gained from the software engineering course."
  - Amos Rweyemamu Mushumbusi
- See also my teaching page !
- The best job in America has a median salary of over $116,000
  - …and likewise in Europe !

Inset: How To Not Apply for Internships

- On 2/14/08, XXX wrote:
  I would kindly ask you therefore if you had any tips for me or could help me in any way. McKinsey & Company is THE company I would love to work for, hence I am very grateful for any support.
- On 2/15/08, XXX Ankur Agrawal wrote:
  Thanks for your email. A couple of questions. First, as you have already applied on the website, did you not hear back from us until now??
  2nd question - how do you expect me to help you without convincing me that you have the potential and that I should recommend you.
- Big misconception you can get @Jacobs: pampering is normal
  - Take your life in your hands!


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

FAQ: What is Software?

- Computer programs + associated documentation such as requirements, design models and user manuals
- Software products may be
  - Generic - developed to be sold to a range of different customers
  - Bespoke (custom) - developed for a single customer according to their specification

FAQ: What is Software Engineering?

- Software engineering =
  the engineering discipline concerned with all aspects of software production
- systematic & organised approach
- appropriate tools & techniques depending on
  - problem to be solved
  - development constraints
  - resources available

FAQ: Difference between *Software Engineering & Computer Science*?

- Computer Science = how computers work, mostly from theoretical & mathematical perspective
  - theory & fundamentals
- Software Engineering = how software systems are built, including project management, quality assurance, and software testing
  - practicalities of developing & delivering useful software
- Both Computer Science & Software Engineering teach fundamentals of programming & computer science
  - can choose either one to become a software developer

[https://www.freecodecamp.org/news/computer-science-vs-software-engineering-which-one-is-a-better-major-88482c38446b/]

FAQ: Difference between *Software Engineering* & *System Engineering*?

- System engineering:
  all aspects of computer-based systems development including hardware, software and process engineering
  - System engineers: involved in system specification, architectural design, integration and deployment
- Software engineering:
  part of this process, concerned with developing the software infrastructure, control, applications and databases in the system

FAQ: What is Good Software?

- Delivers required functionality & performance to the user and is maintainable, efficient, dependable and acceptable
  - Maintainability – can evolve to meet changing needs
  - Efficiency – no wasteful use of system resources
  - Dependability – is trustworthy
  - Acceptability – accepted by the users for which it was designed; understandable, usable and compatible with other systems

FAQ: What Is Software Quality?