TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Software- und Multimediatechnik, Lehrstuhl für Softwaretechnologie

1. Problems of Big Software

Prof. Dr. rer. nat. habil. Uwe Aßmann
Dr.-Ing. Sebastian Götz
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
Technische Universität Dresden
2014-0.1, 07.10.2014

Obligatory Reading

- **Balzert, Kapitel über Entscheidungstabellen**
- **Ghezzi 6.3 Decision-table based testing**
- **Pfleeger 4.4, 5.6**
- **Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. IEEE Transactions on Computers, C-35:677-691, 1986.**
- **Red Hat. JBoss Enterprise BRMS Platform 5: JBoss Rules 5 Reference Guide. (lots of examples for ECA Drools)**
  - http://docs.redhat.com/docs/en-US/JBoss_Enterprise_BRMS_Platform/5/pdf/JBoss_Rules_5_Reference_Guide/JBoss_Enterprise_BRMS_Platform-5-JBoss_Rules_5_Reference_Guide-en-US.pdf

Obligatory Literature

- **Balzert doesn’t contain much**
- **Ghezzi Chapter 1 *or***
- **Pfleeger Chapter 1; Chap 8.1**

- **http://homepages.cs.ncl.ac.uk/brian.randell/NATO/**
  The first International Conference on Software Engineering (ICSE) 1968.

Other Literature

- **S. Garfunkel: Die schönsten Software-Fehler** **http://www.wired.com/news/technology/bugs/0,2924,69355,00.html**
- **Risks.org: www.risks.org Die Seite für Softwarefehler**

What is “Big”?

Class	Lines of Code	Person years
Very small	<1000	<0.2
Small	1000 - 10000	0.2 - 2
Medium	10000 - 100000	2 - 20
Large	100000 – 1 Mio	20 - 200
Very large	>1. Mio	>200

Big Systems

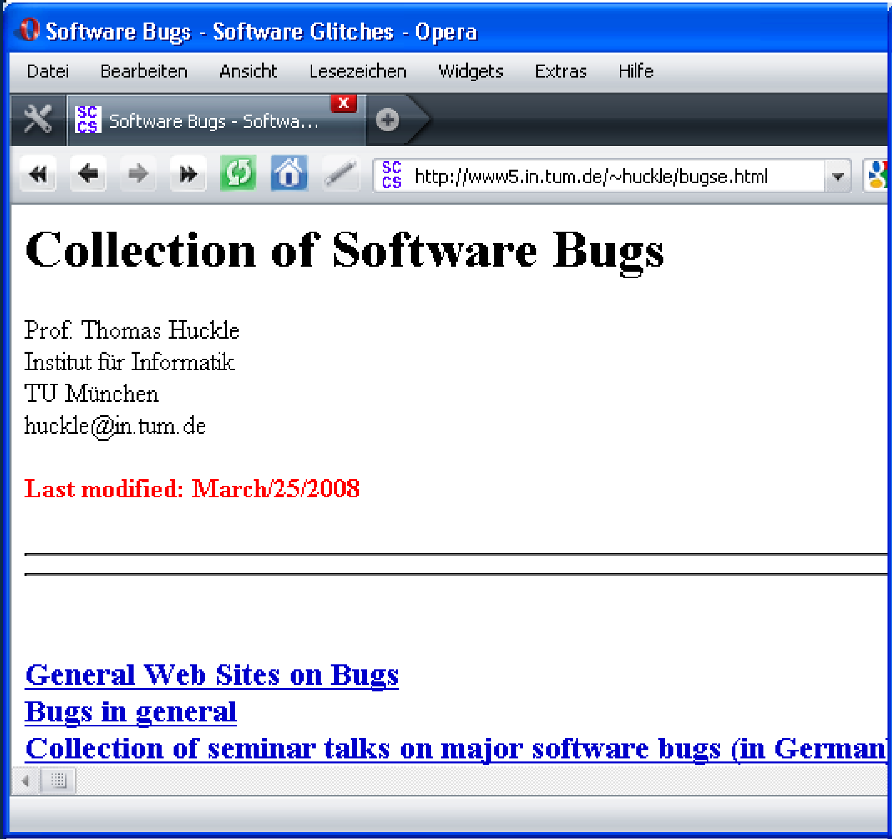
- **Telefone switching software Siemens EWSD (Version 8.1):**
  - 12,5 Mio. lines of code
  - ca. 6000 person years
- **ERP-Software SAP R/3 (Version 4.0)**
  - ca. 50 Mio. lines of code
- **Total amount of lines of code in software (around 2000):**

Credit Suisse	25 Mio. Code-Zeilen
Chase Manhattan Bank:	200 Mio. Code-Zeilen
Citicorp Bank:	400 Mio. Code-Zeilen
AT&T:	500 Mio. Code-Zeilen
General Motors:	2 Mrd. Code-Zeilen

EWSD = Elektronisches Wählsystem Digital (Siemens)
ERP = Enterprise Resource Planning
SAP: Deutscher Software-Konzern

Collection of Software Bugs

- **Web site of Prof. Thomas Huckle**
- **http://www5.in.tum.de/~huckle/bugse.html**

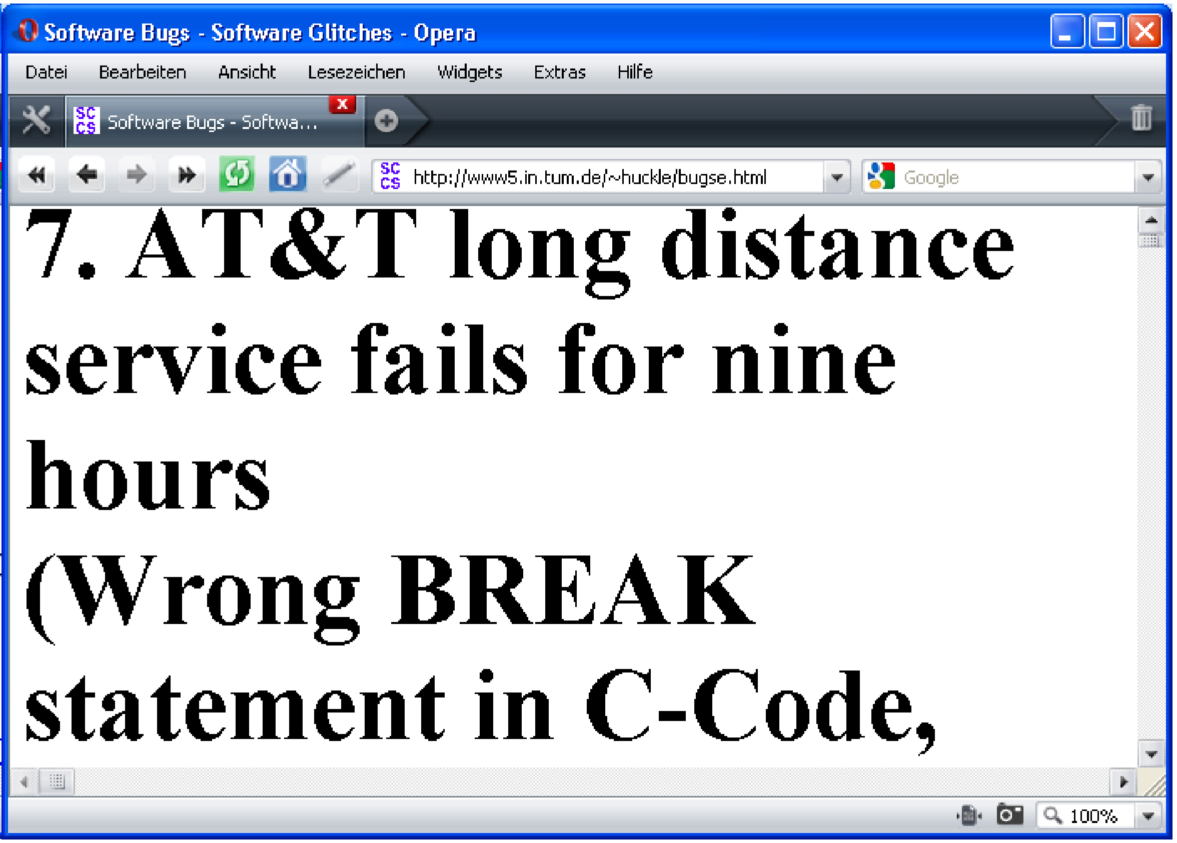

Software Bugs - Software Glitches - Opera
Datei Bearbeiten Ansicht Lesezeichen Widgets Extras Hilfe
Software Bugs - Softwa...
http://www5.in.tum.de/~huckle/bugse.html

Collection of Software Bugs

Prof. Thomas Huckle
Institut für Informatik
TU München
huckle@in.tum.de

Last modified: March/25/2008

General Web Sites on Bugs
Bugs in general
Collection of seminar talks on major software bugs (in German


Software Bugs - Software Glitches - Opera
Datei Bearbeiten Ansicht Lesezeichen Widgets Extras Hilfe
Software Bugs - Softwa...
http://www5.in.tum.de/~huckle/bugse.html
Google
100%

7. AT&T long distance service fails for nine hours
(Wrong BREAK statement in C-Code,

Software Bugs

- **Peter G. Neumann http://www.risks.org**
  The *Risk Digest* collects all possible software bugs

Mercedes console display with conflicting information
<Henry Baker <hbaker1@pipeline.com>>
Fri, 14 Dec 2007 10:48:39 -0800

The console display says "check engine" & "no malfunction" at the same time! Dueling messages!

It is supposed to say "check engine" & "1 malfunction", if "check engine" is the only malfunction being reported.

Permanent Software Crisis?

- **Software Crisis in 1960s:**

We undoubtedly produce software by backward techniques.
D. McIlroy, ICSE 1968

Permanent Software Crisis?

- **Software Crisis (1960s)**
  - Software complexity swiftly increases
  - Costs of software exceeded costs of hardware
  - Techniques to engineer software couldn't handle the ever increasing complexity
- **Software Crisis started in 1968, but exists still today**
  - 70s:
    - Modularity was discovered (as a means to address the complexity issue)
  - 90s, Millennium:
    - Much larger software systems
    - Massive testing necessary

To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

E. W. Dijkstra, ICSE 1968

The Industry

- **Top Players: IBM, Microsoft, HP, Hitachi, Computer Associates, Google, Oracle, SAP**
- **2/3 standard software : 1/3 individual software (with growing rate)**
- **Life Cycle of Software**
  - Average: 5 – 15y
  - max > 35 y (control software, certified systems, data bases)
    - Programmers die out
  - Development time: 1 – 3y

Costs

- **Contrary to Grosch's Law, hardware speed doubles every 2 years, but software productivity increases only about < 5% per year**
- **Costs**
  - acad. Prototype / acad. Product / Product = 1 : 3 : 3
  - Commercialization is rather difficult
- **Relation of development and maintenance 40:60 up to 20:80**
  - Development and maintenance are usually done by different departments
- **Costs: Extreme Requirements**
  - **Certification**: show the software and its development process to a certification agency (TÜV, etc.)
  - **Insurance**: certified software must be executable after 40 years
  - **Example**: German pension rules of the 50s must be processed today
    - Nobody knows the details anymore
    - Solution: write an interpreter for the old assembler
    - This has happened twice..

Cost Example: The Year 2000 Problem

- **COBOL programmers saved space and stored only the last two digits of the year**
  - In the 70s, programs should only live 20 years
- **In 2000, catastrophes were prophesied**
  - Power plants?
  - Pension insurances (birth dates)
- **From 1996 on, the industry panicked**
  - Spent enormous amounts to update software
- **New systems got installed**
  - SAP R/3 with date data type
- **Rewriting didn’t work**
  - Programmers didn’t trust the rewrites
  - Solution: sliding window technique

Cost Example: The Euro Introduction

- **End of 2001, many countries introduced the Euro**
- **Too bad: on paper, the Euro was introduced 2 years before**
  - Some companies had to maintain double booking for 2 years
  - At least for some months in 2002
  - Double booking was very costly: accounts had to be printed in two currencies
- **How to test the transition?**
  - In May 2001, the Dresdner bank ran a test
  - Which failed,.. And produced many wrong money transfers!
- **Many people worked day and night...**

Cost Example: Telecommunication

- **Telecommunication: Failure < 1 h./40 y., working rate 99.999%**
  - One second failure may cost $5Mio
- **Telecommunication software product line**
  - 20-30 000 Module of 1000 loc (lines of code)
  - Single product has 2-8000 modules
- **Necessary: 5000 persons/7years.**
- **Costs ca. 7 billion €.**
- **Size of world market 50 billion €**
- **How many suppliers can exist?**

Human Problems

- **Programmers are not educated well**
  - To develop
  - To communicate
- **Software construction is a social process**
  - It’s hard to organize people
- **Software stays, the people go**
  - Software evolves, many versions coexist and coevolve
- **Projects run out of time**
  - How to control?
- **Programmer Productivity – Rules of Thumb**
  - System software: 1000-2000 loc/y
  - System-like software: 5000 loc/y
  - Application software: 5-10.000 loc/y
- **Individual differences up to factor 5**
  - Has not changed in the last 30 years
- **Differences by programming language and reuse mechanisms**

History of the Term "Software-Engineering"

Softwaretechnologie (Software-Engineering)
Softwareingenieurswesen
Softwaretechnik: Einzeltechnik aus der Lehre der Softwaretechnologie

software engineering: Die Entdeckung und Anwendung solider Ingenieur-Prinzipien mit dem Ziel, auf wirtschaftliche Weise Software zu bekommen, die zuverlässig ist und auf realen Rechnern läuft.

(F.L. Bauer, NATO-Konferenz Software-Engineering 1968)

A Little History

- **NATO Conference on Software Engineering in Garmisch-Patenkirchen. Oct 7-10, 1968**
- **"The whole trouble comes from the fact that there is so much tinkering with software. It is not made in a clean fabrication process. What we need is Software Engineering." Friedrich L. Bauer, 1968**
- **Hence the conference was called "on Software Engineering" [in Thayer&McGettrick IEEE Press]**
- **→ "Software Crisis"**
- **"Component Software"**

Friedrich L. Bauer

Photos of Famous People at ICSE I

E. Dijkstra
"Strukturierte Programmierung"

K. Samuelson
"Stack"

D. McIlroy
"Mass-produced Software Components"

W. van der Poel
"ZEBRA"

B. Randell
"Fehlertoleranz"

D. Gries
"The Science of Programming"

T. Hoare
"Hoare Kalkül"

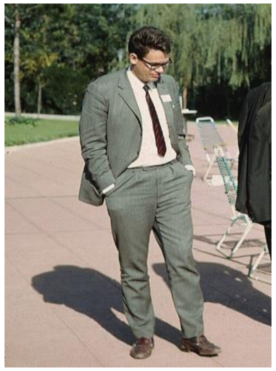
G. Goos (3rd on the right)
"Compiler Construction"

Photos of Famous People at ICSE I

A. Perlis
"Algol",
"Computer Science"

C. Strachey
"denotational semantics"

N. Wirth
"Pascal", "EBNF"


P. Brinch Hansen
"Operating System Principles"

Different Forms of Software – A Classification

- **Artefact: (lat. artificially made)** Code or text or graphics that is made for software (documentation, specification, code, models, etc.)
- **Program:** Sources with object files, libraries
- **Model:** Partial program, abstracting from many details, cannot directly be executed, used during development
- **Software:** Program with user and developer documentation, requirement specification, design descriptions, implementation description, well-elaborated test suite
- **Product:** Mature software. Good, simple, and pedagogic documentation. Simple Installation. Support guaranteed
  - Companies like products
- **Product line (product family):** A group of products, having a common framework and product-specific extensions.
  - Note: every product is sold independently
- **Framework:** A software skeleton for many or all products in a product line

Lehman's Classes of Programs

- **Specification programs (S-programs)**
  - A formal problem specification exists, describing problem and solution
  - The specification allows for checking the solution on validity (formal checks or formal proofs)
- **Problem solving programs (P-programs)**
  - Can be formalized and checked
  - Have requirements for usability and appropriate
- **Embedded programs (E-programs)**
  - Embedded in a social context
  - The specification is a social process; the functionality depends on the involved people
  - No correctness proofs possible

A Software Engineers Career

- **First, you will be a designer and programmer in a team**
  - You will need design skills most urgently for your own and small-size projects
  - In the software process, design flaws are most costly
- **Afterwards, you will be project leader**
  - Without good knowledge in design, you will not be a good developer nor project leader
- **And then manager**
  - But neither a good manager
  - Basic Microsoft strategy: every manager must be able to program
- **.. but some gamble instead [Gates]...**

Your Career (II)

- **Some become entrepreneurs**
- **What is an entrepreneur?**
  - [Prof. R. Würth: Lecture notes on entrepreneurship http://www.iep.uni-karlsruhe.de/seite_260.php]

Ein Unternehmer ist ein Problemlöser.
Insbesondere sind ein Unternehmer und ein Kapitalist zweierlei. Der Kapitalist sieht den Gewinn im Mittelpunkt, aber der Unternehmer findet seine Befriedigung nur im Lösen von Problemen seiner Kunden und seiner Mitarbeiter. Damit kann er zwar auch Geld verdienen, im Wesentlichen lebt er aber nur einen grundlegenden Zug des Menschen aus: für Probleme befriedigende Lösungen zu finden.

What did we learn?

- **Big software creates big problems**
- **Some software has extreme requirements**
- **Sound engineering is necessary**

The End

- **Some german slides are courtesy to Prof. H. Hussmann and Prof. G. Goos.**