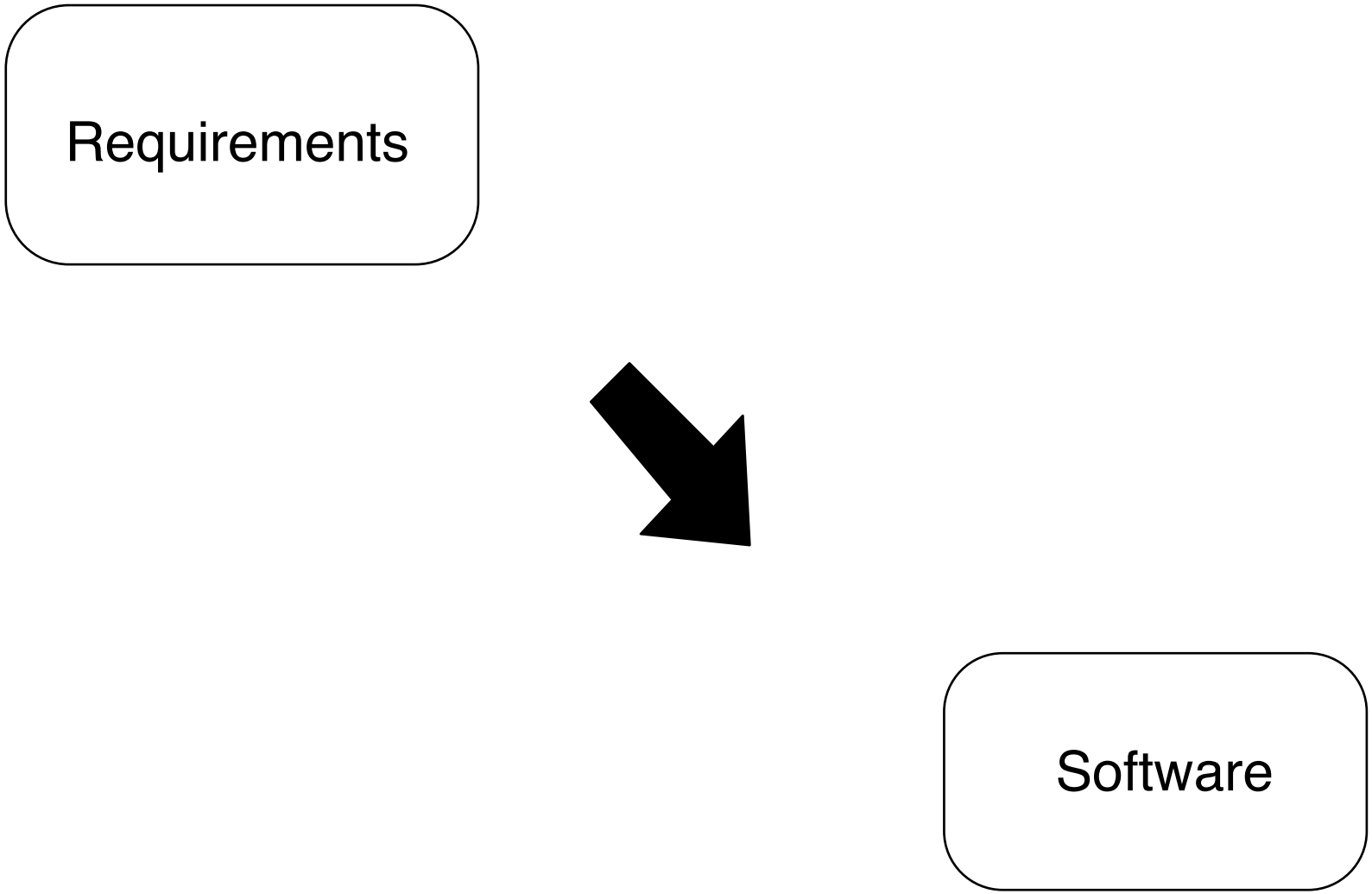
TUM

Software Engineering II
Introduction

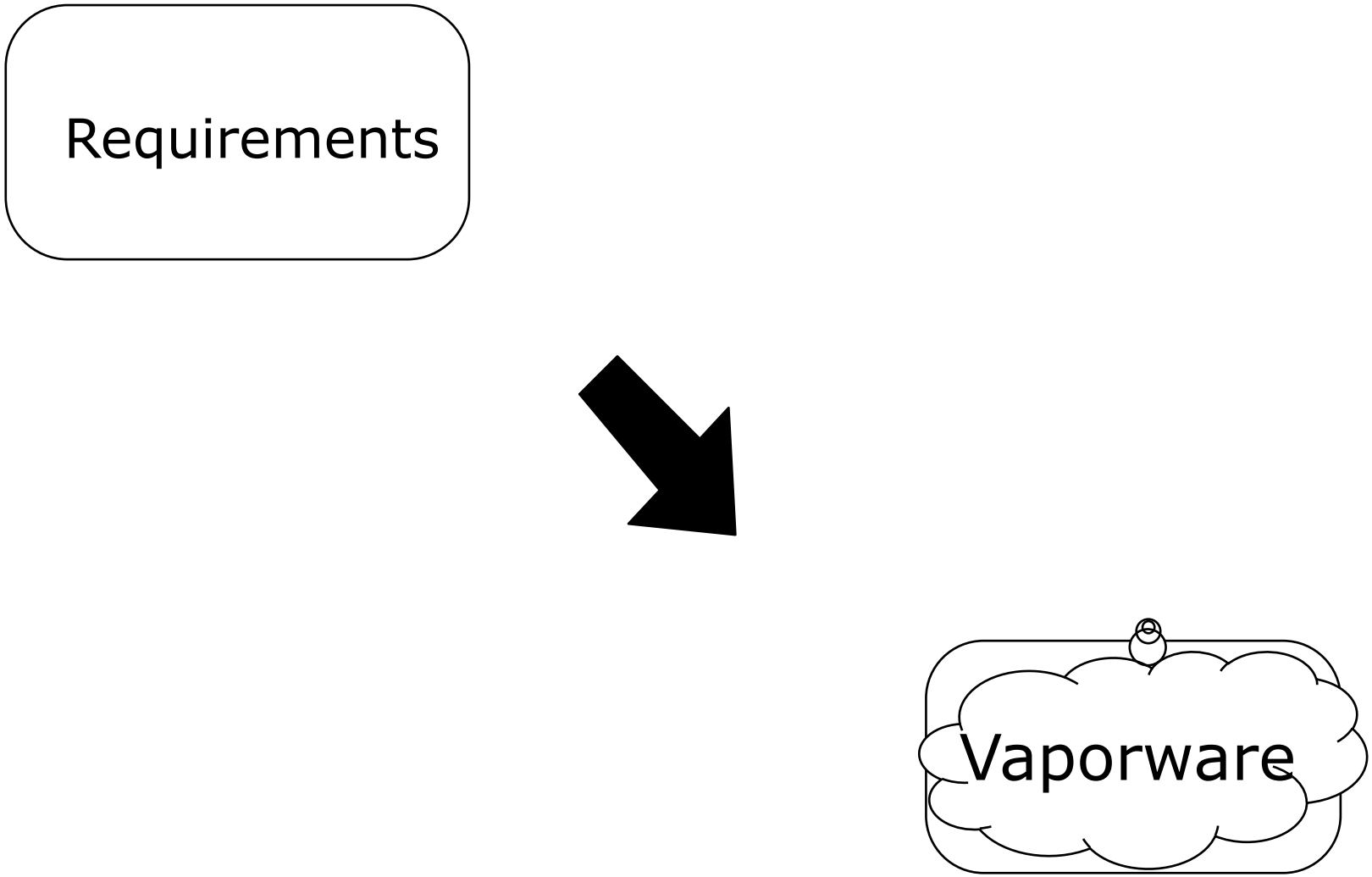
21 April 2009

Bernd Bruegge
Technische Universität München
Institut für Informatik
Lehrstuhl für Angewandte Softwaretechnik
http://wwwbruegge.in.tum.de

What we intend



Limitations of Non-engineered Software



Software Production has Poor Track Record

- **Example:** Space Shuttle Software
- **Cost:** $10 Billion, millions of dollars more than planned
- **Time:** 3 years late
- **Quality:** First launch of Columbia was cancelled
  - **Synchronization problem with the Shuttle's 5 onboard computers.**
- **Substantial errors still exist**
  - **Astronauts are supplied with a book of known software problems "Program Notes and Waivers".**

Toll Collect

- Goal: Toll-collection system for trucks on the autobahn
  - **Government estimate for toll fee income: 156 Mio / month**
  - **In May 2004, 10 new countries will join the European Union, the number of trucks driving across the autobahn will soar, swelling the potential for large fee income.**
- Design:
  - **No tollbooths to slow down traffic.**
  - **Uses a network of satellites, infrared cameras and wireless technology to pinpoint, track and bill trucks for the number of kilometers traveled.**
  - **Drivers install an electronic box on their dashboard, which picks up the satellite signal and transmits data to a central computer.**
  - **Billing is through the mail or over the Internet.**
  - **Modern *reusable* system that could sold to other countries**

Project Plan

- September 2002:
  - **The Ministry of Transport, Building and Housing awards the contract to Deutsche Telekom and DaimlerChrysler.**
- Schedule:
  - **Build, install and test the system within 11 months of signing the contract.**

Actual Project Timeline

- 31 Aug 2003: Project Deadline is missed
  - **New Deadline is agreed upon: 1. November 2003**
- 1 September 2003: First system rollout, a letter to the truck companies lists the following defects:
  - **„Toll free is displayed - although this is not the case“„ “The device does not respond to input“, "Different amounts are charged for the same autobahn segments“, “The device cannot be turned off".**
  - **PS: „The defective devices will be replaced free of charge“**
- 1 October 2003:
  - **DaimlerChrysler and Deutsche Telekom oust Toll Collect's chief executive. Hans-Burghardt Ziermann is the new CEO.**

Comments from Project Stakeholders

- Michael Zirpel, ministry spokesman:
  - **"They were ambitious, saying, 'There is no doubt at all that we will be able to build the system by the end of August'"**
  - **"Only now, when they fail, are they saying, "The timetable is too tight.' Grownup people should know what they signed.'"**
- Dirk Fischer, member of the parliament:
  - **„We have a much more complicated solution, when we should have used a much simpler solution"**
- Rainer Knubben, DaimlerChrysler spokesman:
  - **"It's a P.R. disaster, ok, maybe disaster is too strong a word. It's an image problem, definitely.... for future of public-private partnerships in Germany."**
- Hans-Burghardt Ziermann, Toll-Collect CEO since Oct 2003
  - **„It's innovative, no doubt about it, the trick is to make it work."**
  - **"Everything had to go right the first time to make the deadline,". "History tells you that everything never goes right the first time."**

Project Timeline Continued

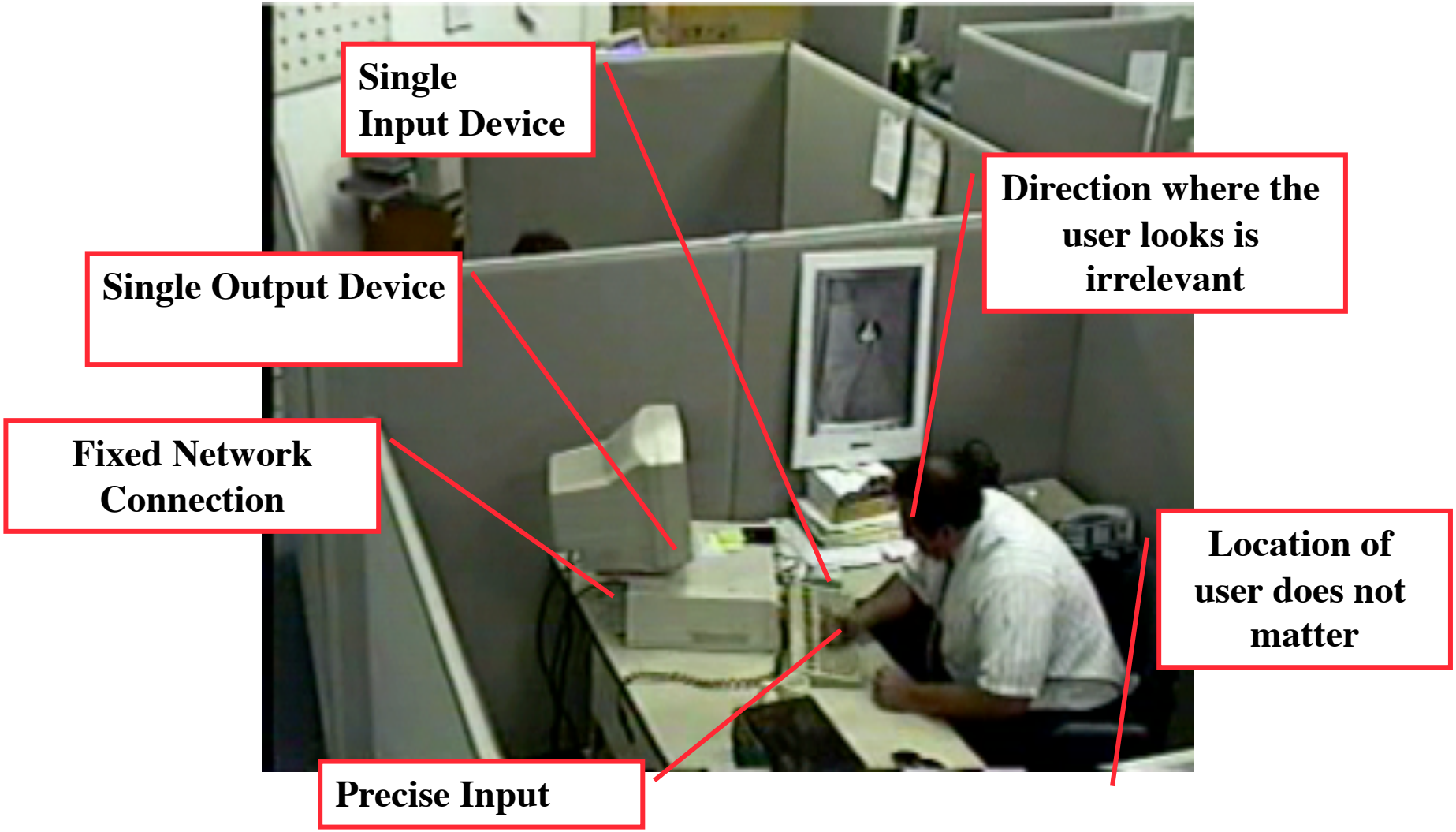
- 1 November 2003: Second deadline is missed
  - **Toll Collect's executives refuse to say when they hope the system will be operational**
  - **Under the terms of the contract, the government can shelve the system by mid December, if it is not running smoothly**
- 1 December 2003:
  - **The companies agree to pay the government 250.000€/day fine until the system is operational**
  - **If the system is not working by March 1, 2004, the fine will double**
  - **Result: Deutsche Telekom and DaimlerChrysler pay fines of 15 million €/month**
- .....
- May 2009 http://www.toll-collect.de
  - **Toll system records and collects charges for more than 100 billion kilometers.**

Quality of today's software….

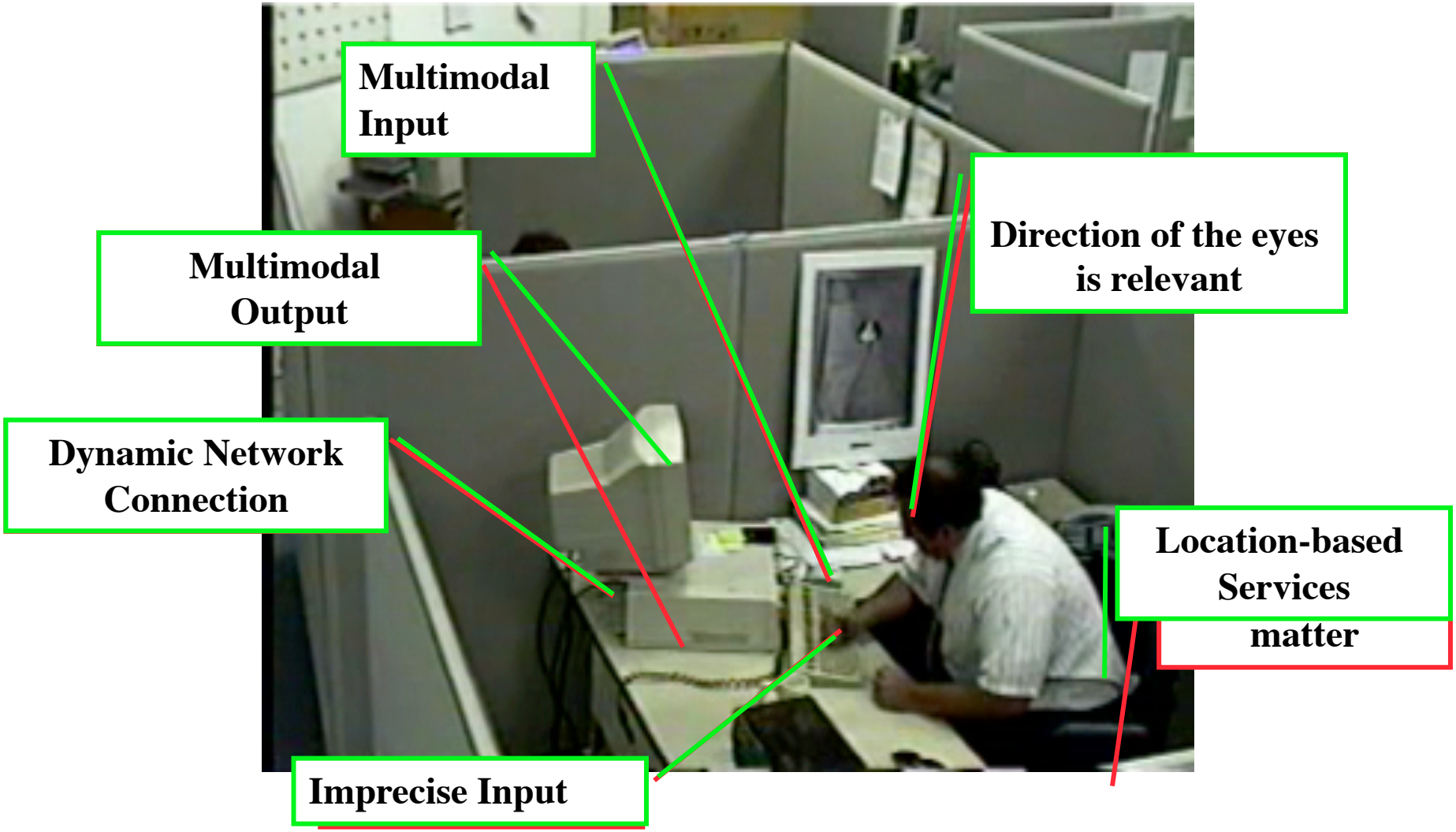
- The average software product released on the market is not error free.

…has major impact on Users

The Old Challenge: Requirements for Desktop-based Systems



The New Challenges: Requirements for Mobile Systems



Typical Parameter in Modern Projects

- The requirements are unknown or hard to elicit from the user
- The requirements change frequently (volatile)
- New technology that effects the outcom appears during the project
- The customer cannot be co-located
- The mobile end user is not able to evaluate visionary scenarios
- We cannot look the end user "over the shoulder"
- The development is distributed.

- What development activities do we need for such a system?
- How do we model such a system?
- How do we elicit requirements from the end user?

Problems in Distributed Development Issues

- Coordination issues
  - **Reduced communication bandwidth**
  - **Increased latency**
  - **Lack of organizational cohesion**
  - **No informal communication**

- Consequences
  - **Changes requests mostly through formal channels**
  - **Changes as 'late surprises'**
  - **Rationale for change request often unavailable.**

Software Engineering: A Problem Solving Activity

- **Analysis:** Understand the problem nature and break the problem into pieces
- **Synthesis:** Put the pieces together into a large structure that prepares for the solution
- Problem solving always happens in the context of methodologies

- **George Polya,** 1887–1985, *How to Solve It,* Princeton University Press, 1957
- Summary of the book:
  - **www.math.utah.edu/~pa/math/polya.html**

Software Engineering: A Problem Solving Activity (2)

- **Methodologies:** Collection of techniques, heuristics and tools unified by a philosphical approach
- **Techniques:** Formal procedures for producing results using some well-defined notation
- **Heuristics:** Informal collection of steps
- **Tools**: Instruments or automated systems that help in accomplishing a technique or supporting heuristics
- Where does management come in?
  - **When the available resources to solve the problem are limited (time, people, budget), or**
  - **If we allow the problem to change**

Methodologies are Paradigms

- Paradigm:
  - **The set of practices that define a scientific discipline *during a particular period of time.***
- Thomas Kuhn, *1922-1996, The Structure of Scientific Revolutions,* 1962
  - **History of science & problem solving.**
  - **Popularized the terms paradigm & paradigm shift**

Software Engineering: Definition

Software Engineering is a collection of techniques, methodologies, tools and heuristics that support the development of

a *high quality software* system

with a given *budget*

before a given *deadline*

while *change* occurs

Example: Running a rapid

A quiet river

Unplanned event: A Hydraulic!

What can we do?

What is Change?

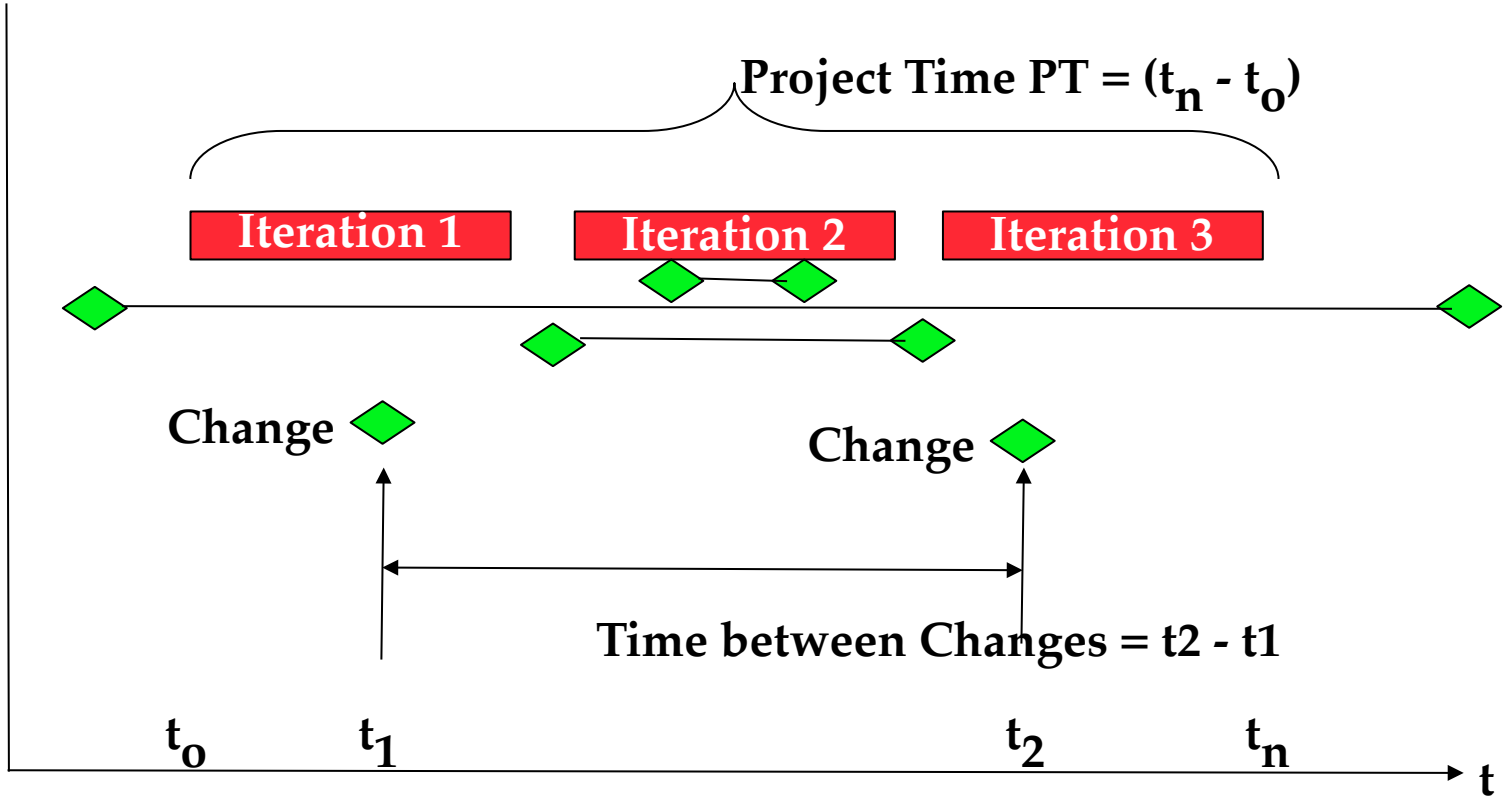
- Change is when something becomes clear ("it crystalizes")
- Change is when something new appears ("technology enabler")
- Change is when something becomes important ("change of requirements")
- Change can happen fast, changes are often unexpected
  - **Manager must anticipate and react to unusual technology**
    - **One strategy: Look at R&D labs: What is done there will be out on the market soon.**
- Hammer and Champy (*Reengineering the Corporation*): "Change is the only thing that is constant".

Examples of Change in Software Development

- Several jobs are combined
  - **Example: Case Worker, a person who deals with the coplete workflow of helping another person**
- Processes have multiple versions to support multiple product
  - **New area: Product line requirements engineering**
- Checks and controls are reduced
  - **Developers have the power to make decisions**
- Work is performed where it makes sense
  - **Outsourcing:** A company subcontracts an activity such as product design or manufacturing, to a third-party company
  - **Offshoring:** A company transfers an organizational function to another country (regardless of whether the work is outsourced or stays within the same company)

Project Duration vs. Rate of Change

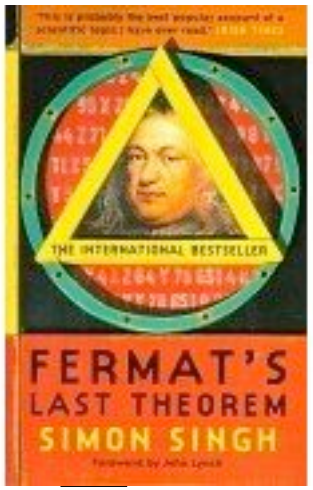
- **PT** = Project Time
- **TPBC** = Time Between Changes (Requirements, Technology)
- **MTBC** = Mean Time Between Changes



How Long Can you Ignore Change?

- Fermat's Last Theorem (1621):
  - **If $n > 2$, $a^n + b^n = c^n$ has no solutions with non-zero integers a, b, and c.**
- Theorem unsolved for 300 years
- Many prizes offered to solve the theorem, most famous the 100.000 Mark prize offered in 1908 by Paul Wolfskehl

- Andrew Wiles, Professor, Princeton University
  - **His "project" from 1984-1993: Trying to solve the theorem without telling anybody**
  - **1993**: Publishes his proof, unfortunately it is wrong
  - **1993-1994:** Repair of the proof (with his former student Richard Taylor)
  - **1997:** Wiles receives the Wolfskehl Prize



Methodology Issues

- Methodology: Provides guidance, general principles and strategies for selecting methods and tools in a given project environment in the context of change
- The key questions for which methodologies typically provide guidance include:

Customer: How much should the customer be involved? Should the team interact with the customer?

Planning: How much planning should be done in advance?

SE I WS 08/09

- **Reuse: How much of the design should result from reusing past solutions?**
- **Modeling: How much of the system should be modeled before it is coded?**

Process: In how much detail should the process be defined?

Project Monitoring: How often should the work be controlled and monitored? When should the project goals be redefined?

Assumptions and Requirements for this Class

- Assumption:
  - **You understand the issues in the analysis or design of a system**
  - **You want to learn more about the managerial aspects of analysis and design of complex software systems**
- Requirements:
  - **You have taken Software Engineering I (Master level, for example SE I, WS 2008/9) or an equivalent class**
- Beneficial:
  - **You have practical experience with maintaining or developing a large software system and have experienced major problems.**

Objectives of the Class

- Appreciate management issues in Software Engineering
  - **Manage the construction complex software systems in the context of frequent change**
- Understand how to
  - **Produce a high quality software system within time**
  - **while dealing with complexity and change**
  - **and create an estimate for the resources needed**
- Appreciate that managerial knowledge is needed when developing software system
- We assume that you have the technical knowledge:
  - **Analysis, Design, Object Design, Implementation**
  - **Modeling with UML.**

Required Technical Knowledge

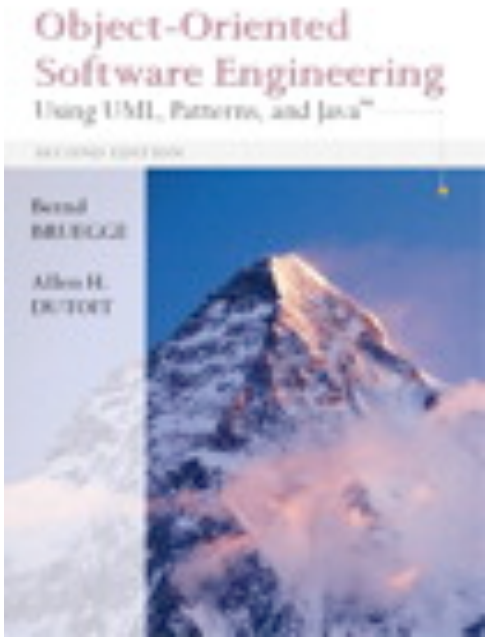
- You understand the problems of system modeling
- You are familiar with UML (Unified Modeling Language)
- You are able to apply different modeling techniques:
  - **Functional modeling, object modeling, dynamic modeling**
- You are able to use design patterns and frameworks
- You understand model-driven architecture (MDA), model-driven design (MDD)
- You understand component-based Software Engineering (CBSE)
- You are able to use a CASE Tool

Intended audience

- Primary Audience:
  - **Students in Informatics (Master, Bachelor with focus on software engineering, Diplom)**
- Secondary Audience
  - **Students who anticipate careers that require significant interaction with software engineers**
- 2+2 SWS, ECTS Credits: 5
- Lecture Time:
  - **Every Tuesday 12:00-14:00, MI 00.08.38 (Multimedia Room)**
  - **First Lecture: Today☺, Last Lecture: 21. July**
- Exercises
  - **Every Wednesday 14:00-16:00, MI 00.08.038**
  - **First Exercise: Wednesday April 22, Last Exercise 8. July**
- Office hours: By appointment.

Reading

- **Textbook: Bernd Bruegge, Allen H. Dutoit, *Object-Oriented Software Engineering; Conquering Complex and Changing Systems*, Prentice Hall, 2003**



- Additional readings are mentioned during each lecture
- Today:
  - **Polya:** *How to Solve it*
  - **Kuhn:** ***The structure of scientific revolutions***
  - **Web Information about Toll Collect**

What do you learn in this class?

- The concept of software lifecycle
  - **Learn about different software lifecycles**
- Different project types
  - **Greenfield Engineering, Interface Engineering, Reengineering**
- The difference between process and product
- Basic project management activities:
  - **Schedule: How to map activities into time**
  - **Organization and People: How to set up a project organization**
  - **Budgeting: How to estimate the cost of a project**
- How to deal with change and uncertainty
- How to set up a project plan, how to control its execution
- How to become a leader, how to deal with teams, and make (Additional literature)
  - **Decisions in the context of project management trade-offs**
  - **Cost vs Schedule, People vs Functionality, Buy vs Build.**

Management vs. Software Project Management

- Management: Getting a specific task done through people
- Management is usually defined in terms of functions:
  - **Planning, organizing, directing, controlling and communicating are typical management functions**
- Project management is defined in the context of a project
  - **Project management tries to accomplish a specific task within a time frame and limited resources.**
- Software project management is defined in the context of a software project
  - **Activities to develop a software system within a given time frame and with limited resources.**
- Think about this distinction as we go through the next slides

Lecture Schedule (Tentative)

- Apr 21: **Introduction**
- Apr 28: **Basic Concepts** (SPMP, IEEE 1058)
- May 5: **Testing** (Managing Tests)
- May 12 : No lecture
- May 19: **Work Breakdown structures** (WBS, Decomposition of work)
- May 26: **Estimation** (Software economics, metrics,traditional planning: Cocomo, Function Points, Agile planning: Planning Poker)
- June 02: No lecture (Pentecost, Pfingsten)
- June 09: **Organization** (team building, customer interaction)
- June 16: Guest lecture (Holger Wolff, Beck et al)
- June 23: **Scheduling** (Project duration, critical path analysis)
- June 30: **Lifecycle Models** (Lifecycle models, IEEE 1074, Unified Process)
- July 7: **Agile Project Management** (XP, Scrum)
- July 14 : Guest Lecture
- July 21: **Knowledge Management** (Acquiring and externalizing knowledge, dealing with conflicts and resolutions)
- To be determined: Exam

Exercise Schedule (Tentative)

- April 22: Icebreaker
- April 29: Software Project Management Plan (SPMP)
- May 6: Software Configuration (SVN, git, SCMP)
- May 13: Testing (J_Unit, Software test plan)
- May 20: Release and Build Management (Cruise Control, Maven)
- May 27: Work Breakdown structures (Create a WBS)
- June 3: Estimation (Establish Estimates)
- June 10: Scheduling (Set up a schedule for a software project)
- **Week of June 15-20: Project Management Day at Accenture (Block Event) Exact date to be announced**
- June 24: Agile Process Management I
- July 1: Agile Project Management II
- July 8: Rationale Management (Unicase)

- You need to do the exercises if you want to be admitted to the exam

Readings

- ***Required Reference***
  - Bernd Bruegge & Allen Dutoit, Object-Oriented Software Engineering: Using UML, Patterns and Java, Prentice Hall 2003. **ISBN 0-13-191179-1** Chapters 11-16
- ***Additional References***
  - IEEE Standards Collection Software Engineering. We cover the following standards
    - [**IEEE Std 1058**], Software Project Management Plan (SPMP)
    - **[IEEE Std 828**] Software Configuration Management
    - **[IEEE Std 1042**] Guide to Configuration Management Plan (SCMP)
    - **[IEEE Std 1074**] Software Lifecycle Management
    - [**IEEE Std. 1074.1**] Guide to IEEE 1074
- **More references are given at the end of each lecture.**

Resources

- Labs (5 computers per lab reserved for class)
  - **Aquarium**
  - **Terrarium**
- Meeting rooms
  - **Glas room**
  - **Seminar room**

Where to find more Information

- Lecture Portal:
  - **http://wwwbruegge.informatik.tu-muenchen.de/twiki/bin/view/Lehrstuhl/POMSS09**
- Contains
  - **Lecture Schedule**
  - **Presentations**
  - **Bibliography**