Einführung in die Softwaretechnik
Introduction into Software Engineering

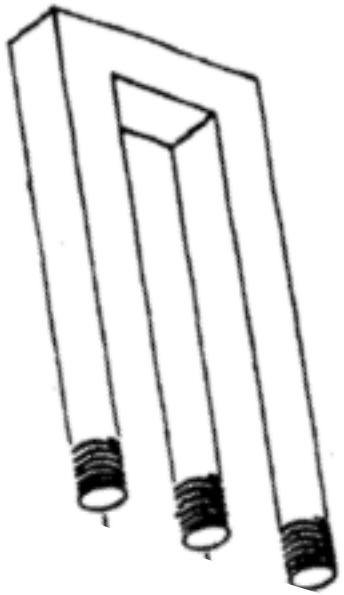
EIST

Introduction into Software Engineering

Bernd Brügge
Lehrstuhl für Angewandte Softwaretechnik
Technische Universität München
14 April 2016

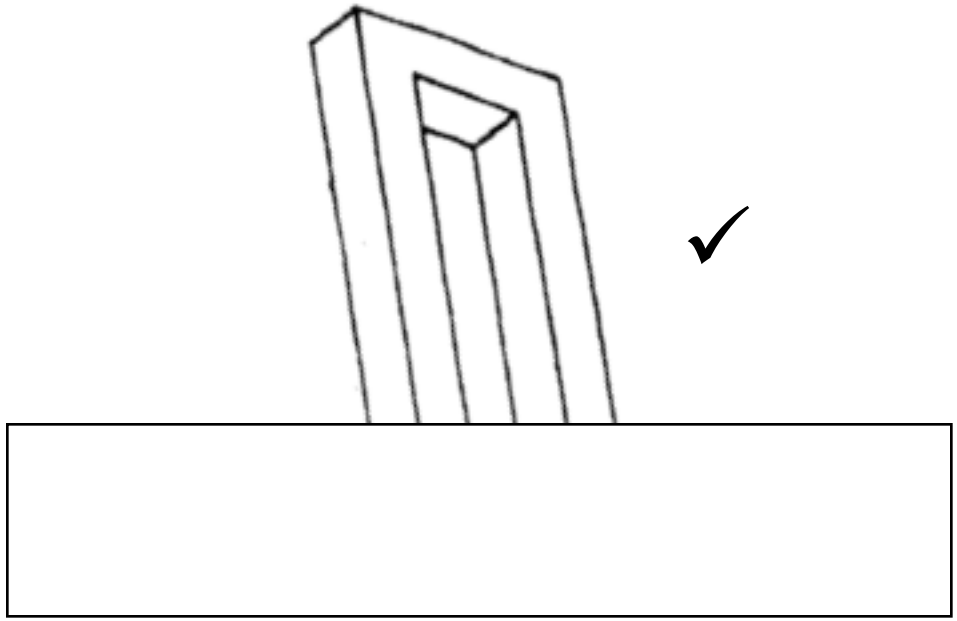
Impossible?
Possible,
But hard?
Can you climb this
Mountain?

Can you develop this System?

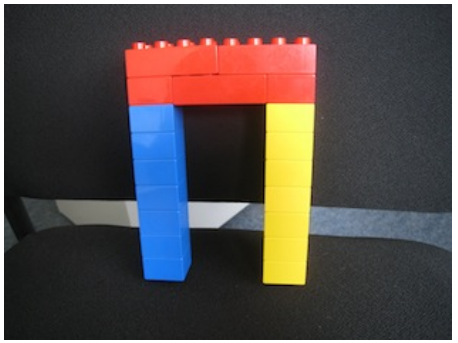


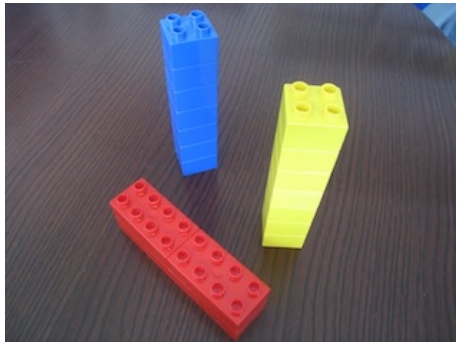
Impossible?
Or only hard?

How about this System?

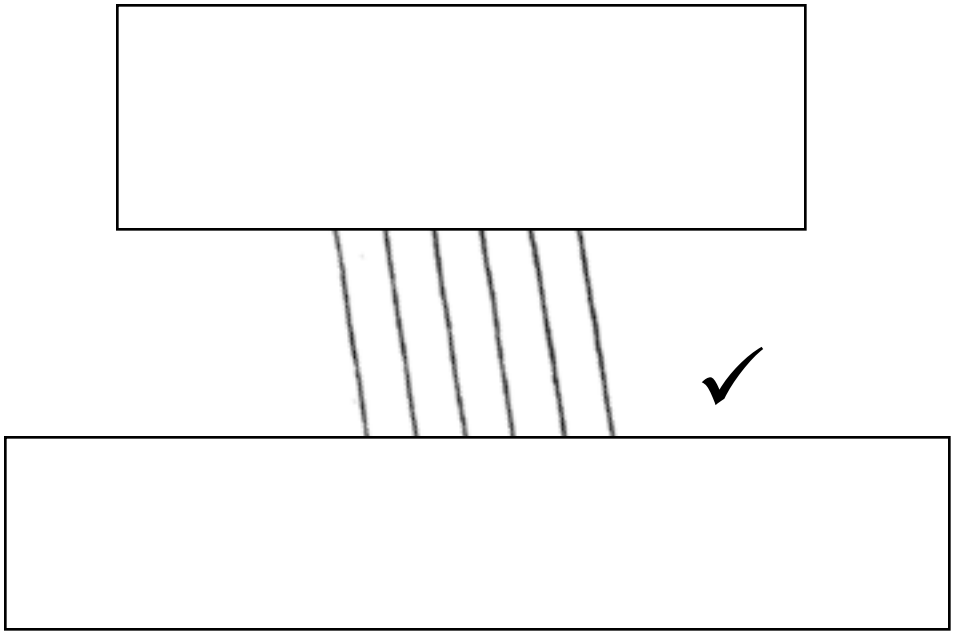


✓

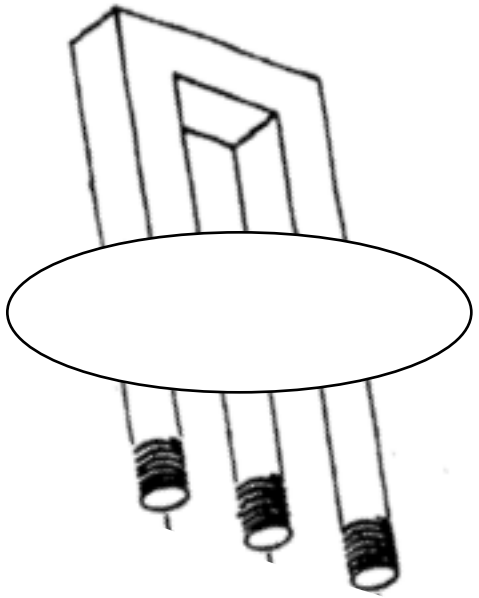




Can you develop this System?

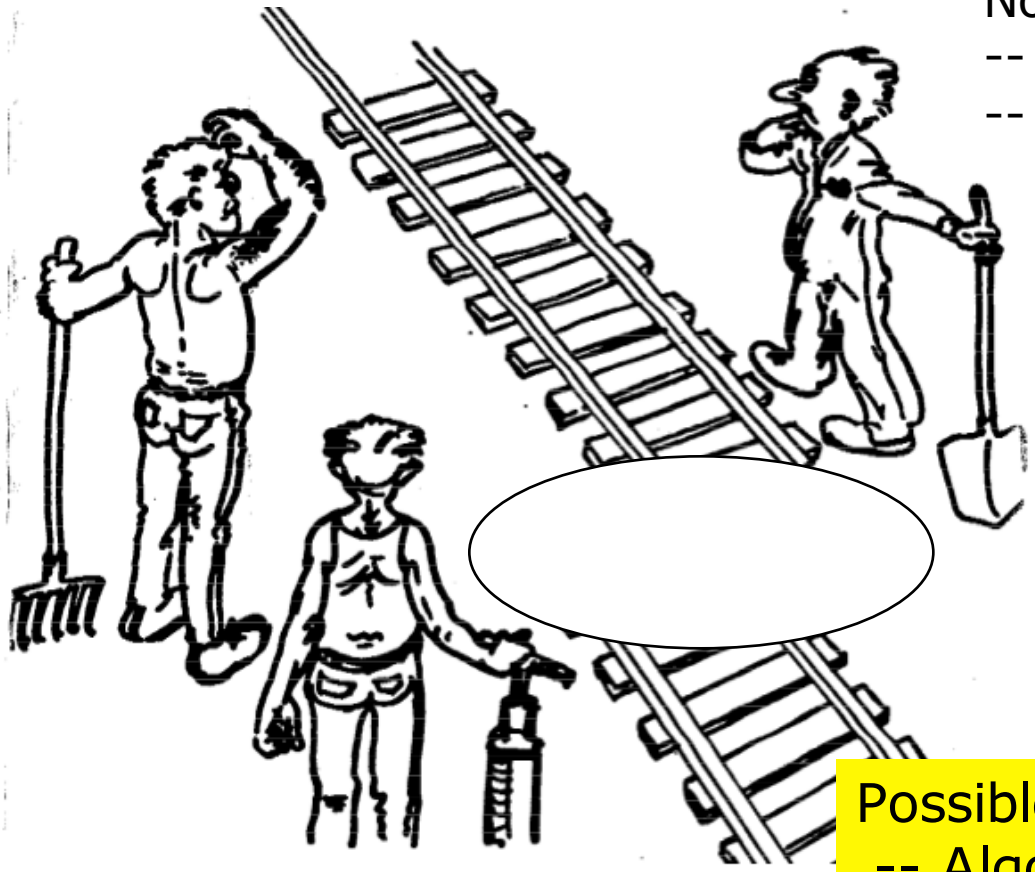


The impossible Fork



The impossibility shows up,
when we physically try
to merge the subsystems from
the previous slides
(during "system integration")

Another System Integration Example



Not impossible!
-- A system integration failure!
-- Or a development problem

Possible reasons:
-- Algorithmic fault
-- Wrong usage of compass
-- Bad communication between teams

Physical Model of the impossible Fork

Shigeo Fukuda, 1932-2009

Impossible Column

http://www.illusionworks.com/mod/fukuda.htm

http://im-possible.info/english/articles/trident/trident.html

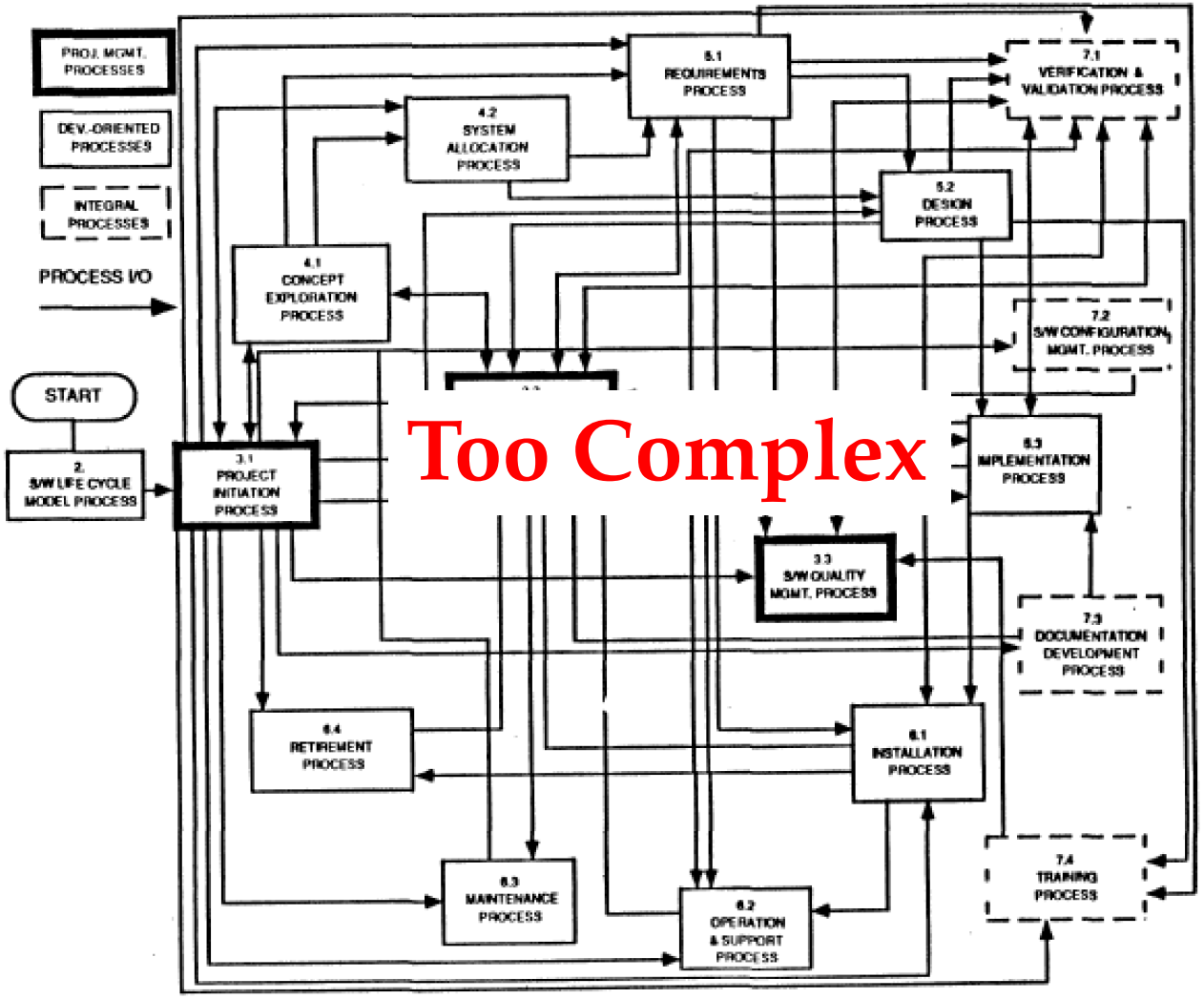
First Insights

1. You need to talk to the customer about the problem scope and the requirements
2. We solve a complex problem by dividing it into smaller pieces ("divide and conquer")
3. We then try to put the pieces back into a larger system that solves the problem
4. Problem descriptions that don't have unique objects cannot be built without reformulating (changing) the problem description.

Software Development is more than just Writing Code

- It is problem solving
  - Understanding the problem
  - Proposing a solution and plan
  - Engineering a system based on the proposed solution using a *good* design
- It is about dealing with **complexity**
  - Creating abstractions and models
  - Notations for abstractions
- It is about dealing with **change**
  - Requirements elicitation, analysis, design, implementation, validation of the system, delivery and maintenance.

What is the problem with this Drawing?



Abstraction

- Complex systems are hard to understand
  - The 7 +- 2 phenomena
    - Our short term memory cannot store more than 7+-2 pieces at the same time -> limitation of the brain
    - My Phone Number

Abstraction

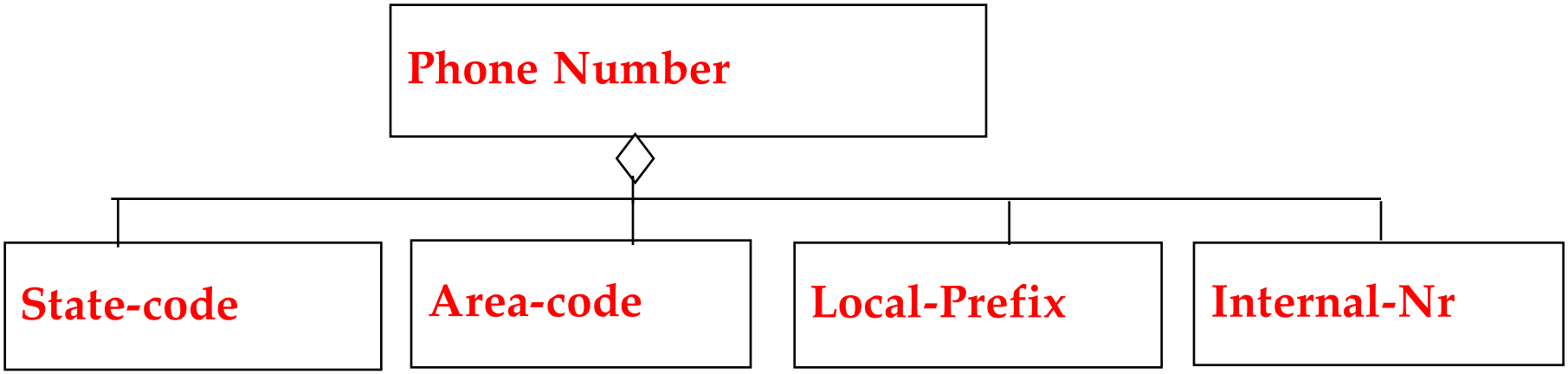
- Complex systems are hard to understand
  - The 7 +- 2 phenomena
    - Our short term memory cannot store more than 7+-2 pieces at the same time -> limitation of the brain
    - My Phone Number

- Chunking:
    - Group collection of objects to reduce complexity
    - 4 chunks:
      - State-code, Area-code, Local-Prefix, Internal-Nr

Abstraction

- Complex systems are hard to understand
  - The 7 +- 2 phenomena
    - Our short term memory cannot store more than 7+-2 pieces at the same time -> limitation of the brain
    - My Phone Number

- Chunking:
    - Group collection of objects to reduce complexity
    - State-code, Area-code, Local-Prefix, Internal-Nr

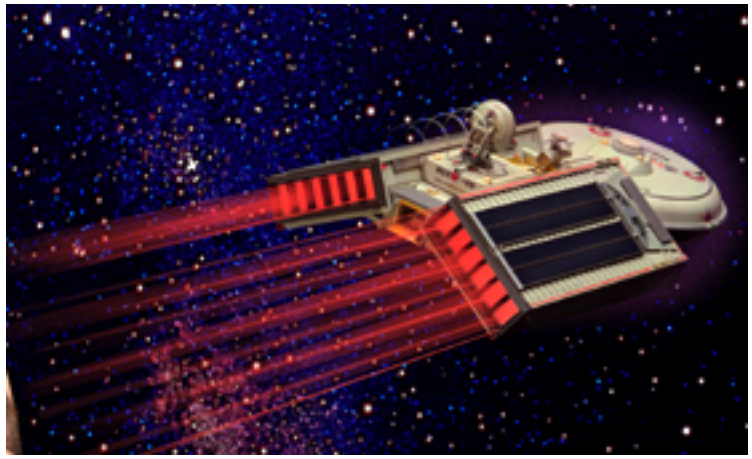


Abstraction

- Abstraction allows us to ignore unessential details
- Two definitions for abstraction:
  - Abstraction is a *thought process* where ideas are distanced from objects
    - **Abstraction as activity**
  - Abstraction is the *resulting idea* of a thought process where an idea has been distanced from an object
    - **Abstraction as entity**
- Ideas can be expressed with a model

Models

- A model is an abstraction of a system
  - A system that no longer exists
  - An existing system
  - A future system to be built.



We use Models to describe Software Systems

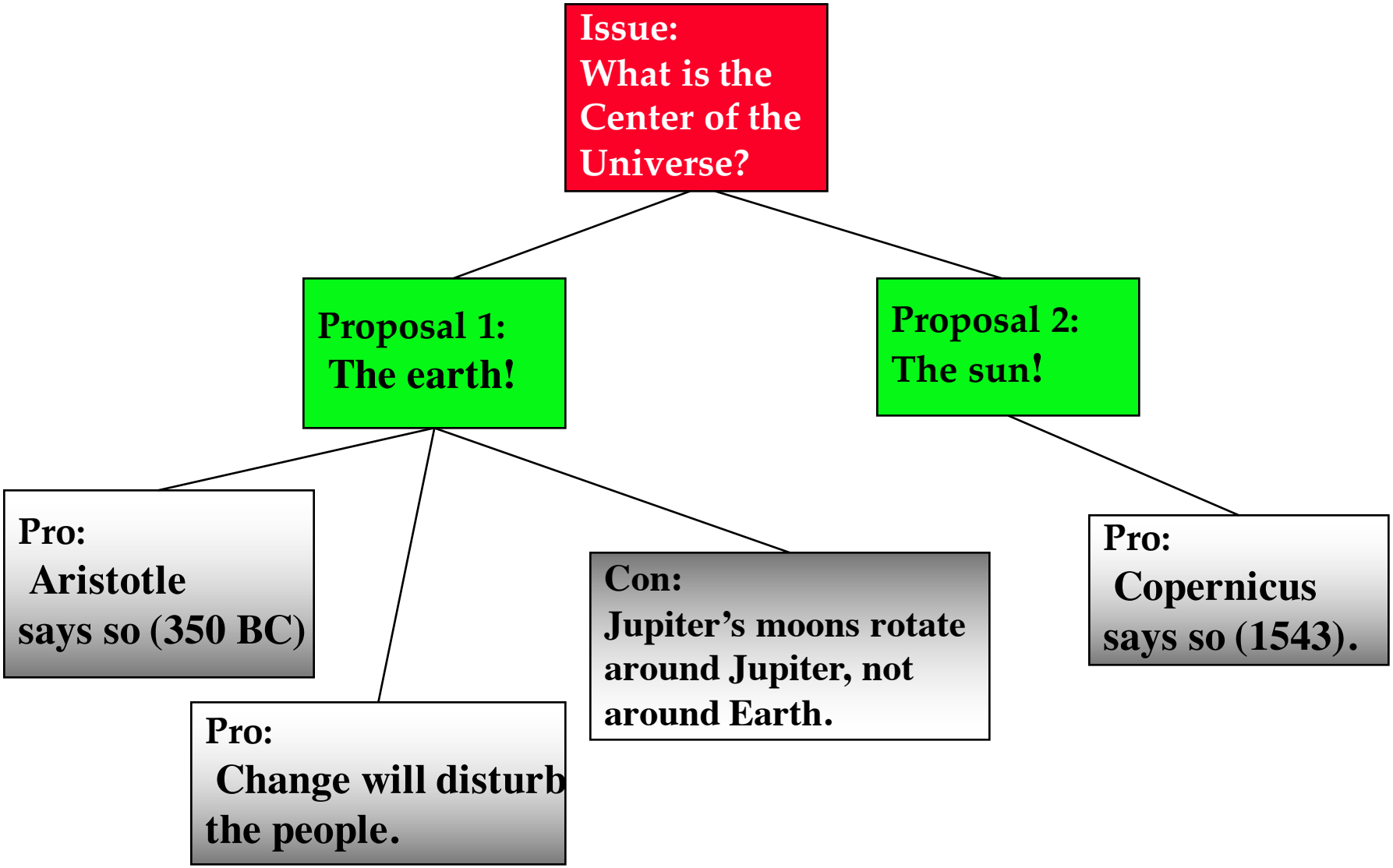
- Object model: What is the structure of the system?
- Functional model: What are the functions of the system?
- Dynamic model: How does the system react to external events?

- System Model: Object model + functional model + dynamic model

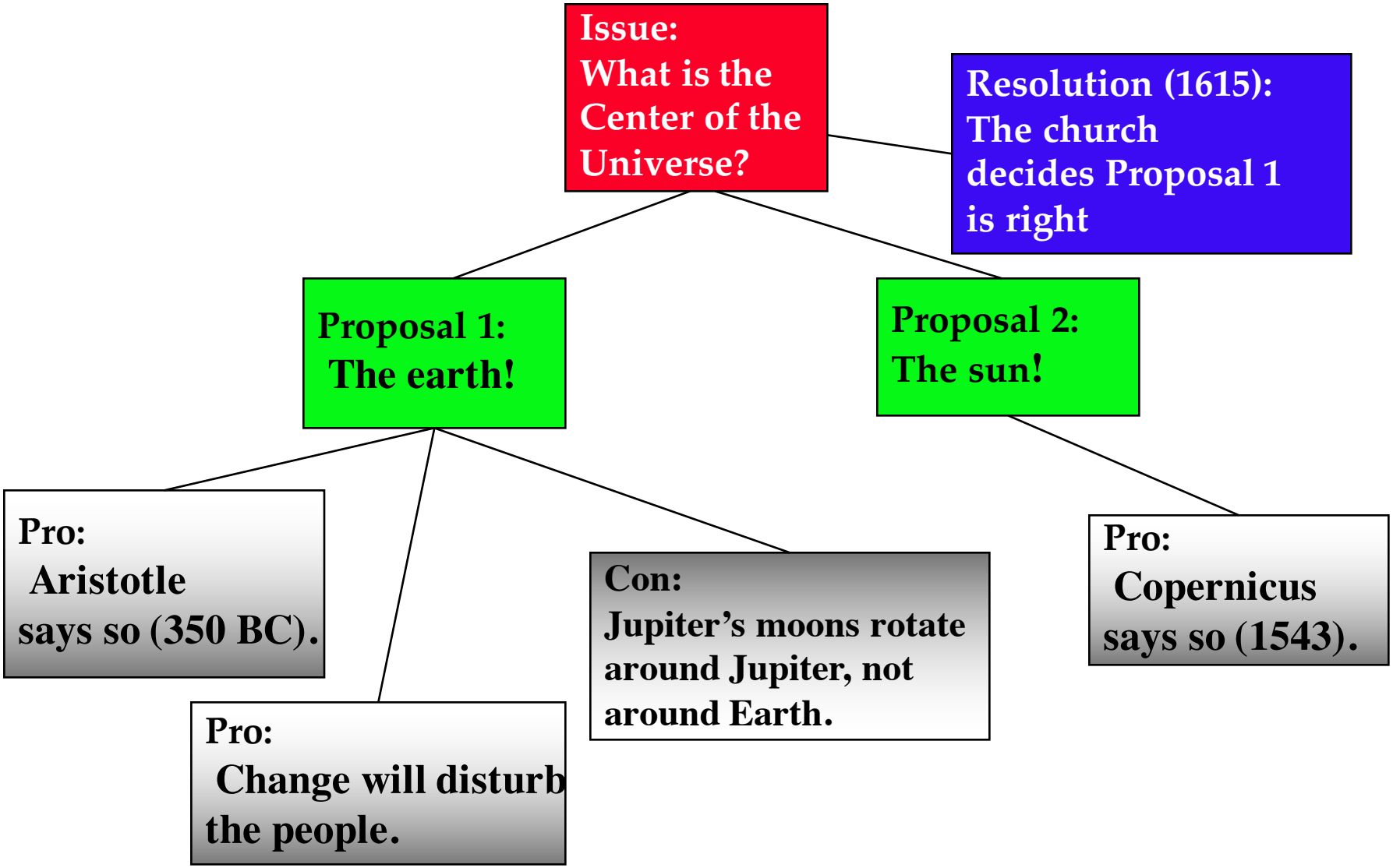
Other Models used to describe Software System Development

- Task Model:
  - PERT Chart: Describes the dependencies between development activities
  - Schedule: Describes how the activities be accomplished by the end of the project
  - Organization Chart: Describes the roles of the participants in the project
- Issue Model:
  - What are the open issues?
  - What proposals do we have to close the issues?
  - What resolutions can be made?

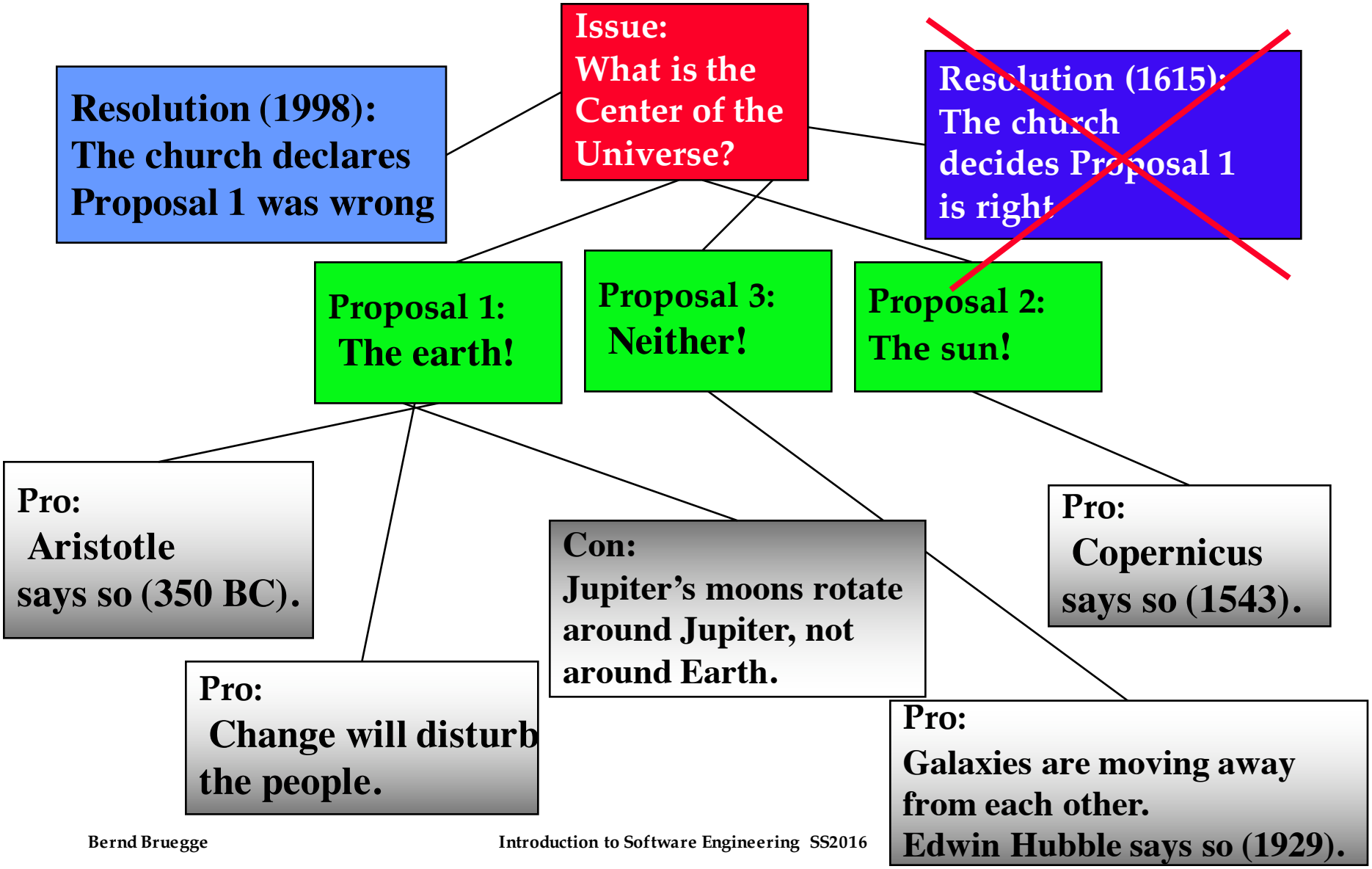
Issue-Modeling



Issue-Modeling



Issue-Modeling



Why is Software Development difficult?

- The problem is usually ambiguous
- The requirements are usually unclear and changing when they become clearer
- The problem domain (also called application domain) is complex, and so is the solution domain
- The development process is difficult to manage
- Software offers extreme flexibility
- Software is a discrete system
  - Continuous systems have no hidden surprises
  - Discrete systems can have hidden surprises!

David Lorge Parnas - an early pioneer in software engineering who formulated in 1972 the concepts of modularity and information hiding which are the foundation of object oriented methodologies.

Software Engineering: A Problem Solving Activity

- **Analysis:**
  - Understand the nature of the problem and break the problem into pieces
- **Synthesis:**
  - Put the pieces together into a larger structure

For problem solving we use techniques, methodologies and tools.

Techniques, Methodologies and Tools

- **Techniques:**
  - Formal procedures for producing results using some well-defined notation
- **Methodologies:**
  - Collection of techniques applied across software development and unified by a philosophical approach
- **Tools:**
  - Instruments or automated systems to accomplish a technique
  - Integrated Development Environment (IDE)
  - Computer Aided Software Engineering (CASE)

Computer Science vs. Engineering

- The computer scientist
  - Assumes techniques and tools have to be developed
  - Proves theorems about algorithms
  - Designs languages and grammars
  - Has infinite time (in principle☺)
- The engineer
  - Develops a solution for a problem formulated by a client
  - Uses computers & languages, techniques and tools
- The software engineer
  - Works in multiple application domains
  - Has only limited time …
  - …while changes occurs in a complex problem formulation and often also in the available technology.

Software Engineering Definition

Software Engineering is a collection of techniques, methodologies and tools that help with the production of

A high quality software system developed with a given *budget* before a given *deadline* while *change* occurs

Challenge: Dealing with complexity and change

Course Outline

Dealing with Complexity

- Modeling
- Notations (UML, OCL)
- Requirements Elicitation
- Analysis and Design
  - OOSE, scenario-based design, formal specifications
- Implementation
- Testing

Dealing with Change

- Release Management
  - Configuration Management
  - Continuous Integration
- Delivery
  - Continuous Delivery
- Software Life Cycle Modeling
- Rationale Management
- Project Management

Course Outline

Dealing with Complexity

- Modeling
- Notations (UML, OCL)
- Requirements Elicitation
- Analysis and Design
  - OOSE, scenario-based design, formal specifications
- Implementation
- Testing

Dealing with Change

- Release Management
  - Configuration Management
  - Continuous Integration
- Delivery
  - Continuous Delivery
- Software Life Cycle Modeling
- Rationale Management
- Project Management

Exercises – the traditional way

- Exercises consist of two parts:
  - **Central Exercises** are done in a central exercise session (Zentralübung) or with tutor classes. The goal is to work on a set of problems and solve them, possibly with the help of the tutor.
  - **Homeworks** to be solved within a week at home

Exercises – the EIST way

- Exercises are part of each lecture:
  - They are intertwined with the lecture
  - There will be no homeworks

Teaching Methodology

- We do not distinguish between lectures and exercises
- We have 2 time slots:
  - Tuesday and Thursday
- In each time slot…

  …we teach you one or more concepts

  … we ask you to do apply these concepts in exercises

➢ Make sure to come with your laptop to class.

- Different names for this type of teaching:
  - Experiential learning
  - Blended Learning
  - Just in Time Learning
  - Continuous Learning
  - Chaordic Learning

Organization of the exercises

- The exercise master: Jan Knobloch

- The Tutors: 24 Students

Linus, Georg, Carolin, Leon, Korbinian, Enrico, Ferdinand, Jonas, Rajat, Lena, Anastasia, Jonathan, Katharina, Paul, Tobias, Michael, Simon, Sonja, Michael, Benedikt, Christian, Matthias, Ray, Maximilian.

What happens if I don't participate in the exercises?

Assumptions for this Course

- You know about Java and object-oriented programming constructs
- You have solved small programming projects
- For example, you have taken:
  - **IN0001** Introduction to Informatics 1
  - **IN1501** Fundamentals of Programming
- Beneficial:
  - You have had practical experience with a large software system
  - You have already participated in a large software project
  - You have experienced major problems in developing a software system.

Objectives of the Class

- Appreciate the Fundamentals of Software Engineering
  - Modeling
  - Project organization
  - Software lifecycle models
  - Requirements engineering
  - Analysis
  - System design and architecture
  - Detailed software design
  - Interface specification
  - Implementation
  - Testing (unit testing, integration testing, system testing)
  - Configuration management
  - Build and release management
  - System maintenance.

Course Schedule (Preliminary)

Lecture	Day	Date	Subject	Time
1	Thu	14.04.2016	Introduction	08:00-10:15
2	Tue	19.04.2016	Modeling with UML #1	12:15-14:00
3	Thu	21.04.2016	Modeling with UML #2	08:00-10:15
4	Tue	26.04.2016	Requirements Elicitation	12:15-14:00
5	Thu	28.04.2016	System Modeling	08:00-10:15
6	Tue	03.05.2016	Design Patterns I and II	12:15-14:00
7	Thu	05.05.2016	Reverse Engineering	08:00-10:15
8	Tue	10.05.2016	Detailed Design	12:15-14:00
9	Thu	12.05.2016	System Design I	08:00-10:15
10	Tue	17.05.2016	System Design II	12:15-14:00
11	Thu	19.05.2016	Quality Management I	08:00-10:15
12	Tue	24.05.2016	Quality Management II	12:15-14:00
13	Thu	26.05.2016	Interface Specification	08:00-10:15
14	Tue	31.05.2016	Model Transformations I	12:15-14:00
15	Thu	02.06.2016	Model Transformations II	08:00-10:15
16	Tue	07.06.2016	Lifecycle Modeling I	12:15-14:00

Course Schedule (Preliminary)

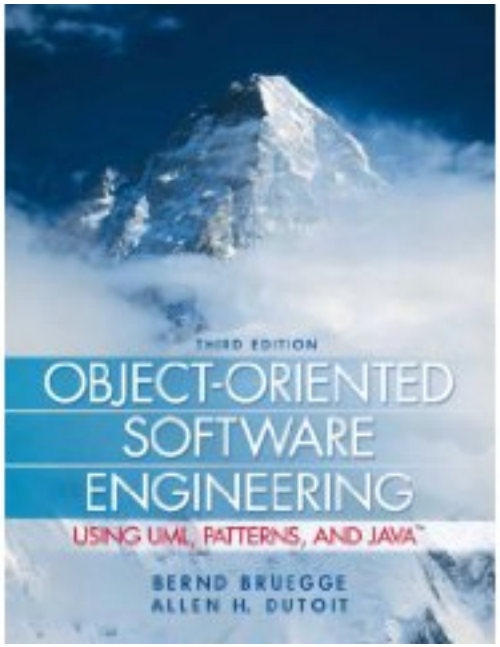
Lecture	Day	Date	Subject	Time
17	Thu	09.06.2016	No Lecture	08:00-10:15
18	Tue	14.06.2016	No Lecture	12:15-14:00
19	Thu	16.06.2016	Guest Lecture	08:00-10:15
29	Tue	21.06.2016	Lifecycle Modeling II	12:15-14:00
21	Thu	23.06.2016	Mapping UML	08:00-10:15
22	Tue	28.06.2016	Presentation of Large complex system	12:15-14:00
23	Thu	30.06.2016	Project Management	08:00-10:15
24	Tue	05.07.2016	Release Management	12:15-14:00
25	Thu	07.07.2016	System Maintenance	08:00-10:15
26	Tue	12.07.2016	Guest Lecture	12:15-14:00
27	Thu	14.07.2016	Wrap-up and Exam Preparation	08:00-10:15

- Final exam: To be announced
- Note: There will be no midterm.

Times and Locations

- Main lecture & Exercises combined:
  - Tuesday 12:15 – 14:00
    - 3 Rooms: FLB Lecture Hall, Small Interims Lecture Hall and Lecture Hall 2
  - Thursday 08:00-10:15 (you are here☺)
    - 2 Rooms: FLB Lecture Hall and Large Interims Lecture hall
  - For the exercises you need register for this course via TUM Online
    - Why? We are going to create accounts for you at our departments infrastructure for submission
- We are filming in the FLB Lecture Hall and transmit the lecture via video streaming into the other two lecture halls.
- Tutors will be in each of the lecture halls to help you with the exercises.

Textbook



Bernd Bruegge, Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns and Java, 3rd Edition

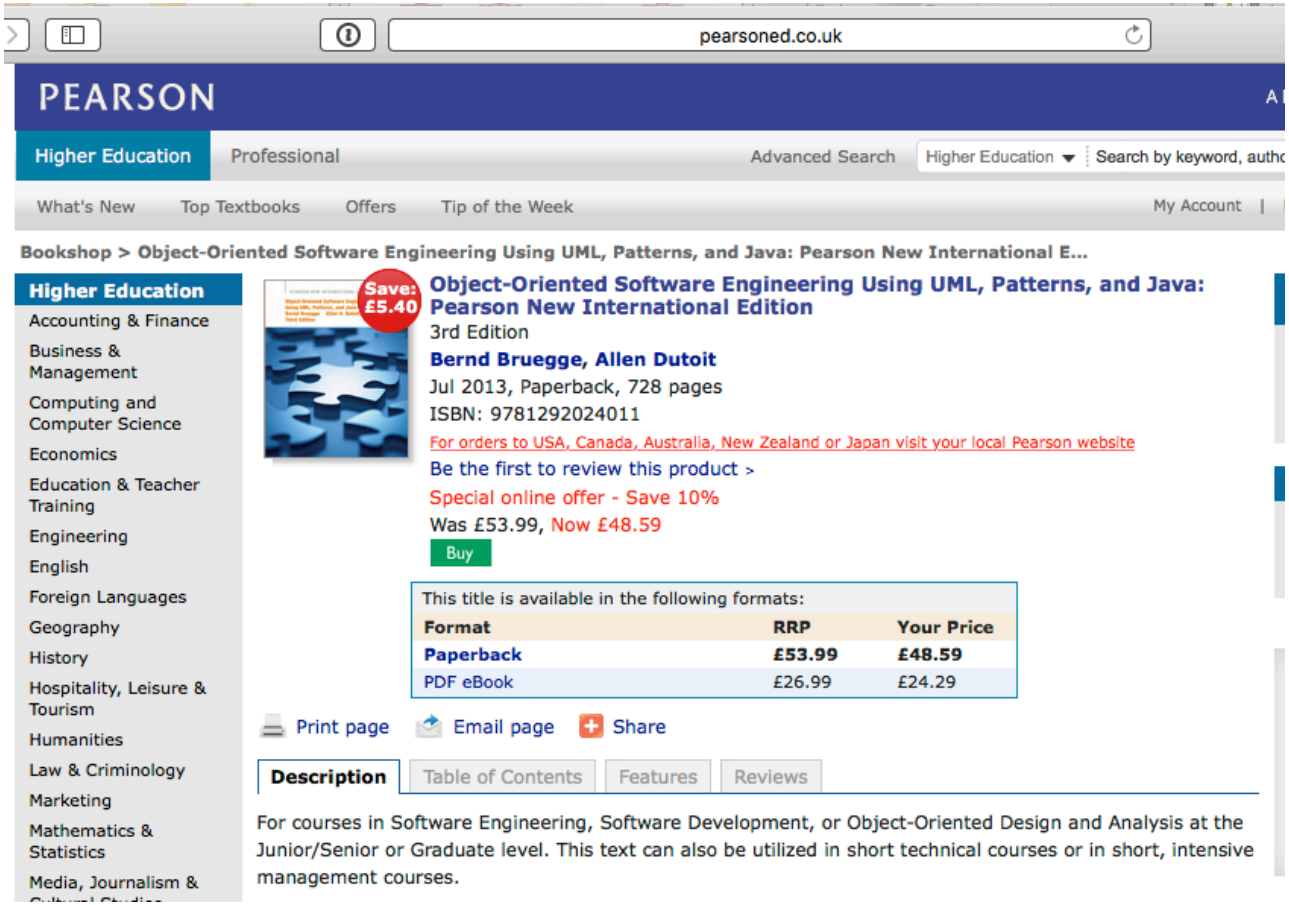
Publisher: **Prentice Hall**, Upper Saddle River, NJ, 2009;

ISBN-10: 0136061257

ISBN-13: 978-0136061250

- Additional readings will be added during each lecture.

International Edition and eBook Versions available

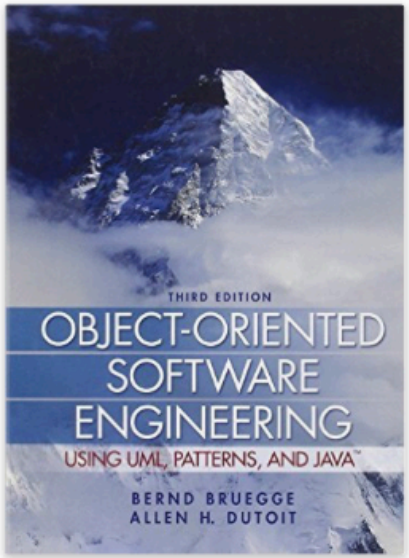


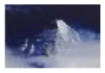
Used copies are also available online…

…but be aware of ridiculous offers

Bücher Erweiterte Suche Stöbern Bestseller Neuheiten Hörbücher Fremdsprachige Bücher Taschenbücher Fachbücher Schulbücher Angebote

‹ Zurück zu den Suchergebnissen für "brügge dutoit"





Dieses Bild anzeigen

Object-Oriented Software Engineering Using UML, Patterns, and Java (3rd Edition) by Bernd Bruegge (2009-08-08)

Gebundene Ausgabe – 1656

von Bernd Bruegge; Allen H. Dutoit (Autor)

Geben Sie die erste Bewertung für diesen Artikel ab

Alle Formate und Ausgaben anzeigen

Gebundene Ausgabe ab EUR 512,06

1 neu ab EUR 512,06

Hinweis: *Dieser Artikel ist nur bei Drittanbietern erhältlich (alle Angebote anzeigen).*

DER SPIEGEL Spiegel Bestseller

Die Spiegel-Bestseller

Entdecken Sie die Bestseller des SPIEGEL-Magazins aus unterschiedlichen Bereichen. Wöchentlich aktualisiert. Hier klicken

Empfehlen

1 neu ab EUR 512,06

Alle Angebote

Auf die Liste

Möchten Sie verkaufen?

Bei Amazon verkaufen

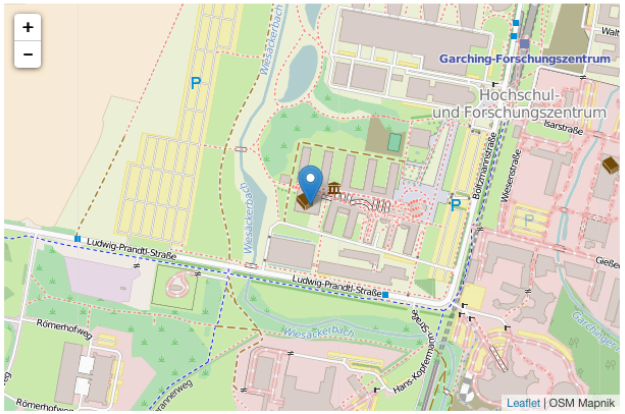
Book Availability On Campus

- Copies in the MI library
  - https://www.ub.tum.de/en/branch-library-mathematics-informatics

- Additional copies in the mechanical engineering library
  - https://www.ub.tum.de/en/branch-library-mechanical-engineering

Branch Library Mathematics & Informatics

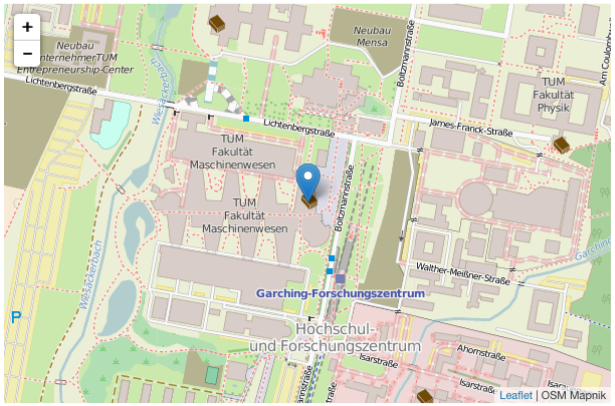
Boltzmannstraße 3
85748 Garching



Email: information@ub.tum.de
Telephone: 089-289-16900

Branch Library Mechanical Engineering

Boltzmannstraße 15
85748 Garching
Fakultätsgebäude Maschinenwesen
Gebäudeteil 0, 1. OG



Email: information@ub.tum.de
Telephone: 089-289-16368

Grading Criteria

- To pass this course, your exam grade must be 4.0 or better
- You can improve your exam grade with a 0.3 bonus by regular participation in the exercises
- For each exercise:
  - we determine the 3 fastest and correct submissions
  - we do a random drawing of all submitted exercises.

EIST Forum

- We will use a class-specific forum where you can communicate with each other about EIST
- The forum is based on Moodle
- The EIST forum will contain the following categories
  - Announcements
    - The schedule (a "sticky" post)
    - Schedule changes, guest lectures
  - Discussions
    - Lecture-specific discussions
  - Lectures
    - Lecture slides (PDF), Lecture movies (MP4)
  - Exercises

    Exercise slides and sample solutions

➢ More information in the next lecture (Tuesday, April 19)

Summary

- Software engineering is problem solving
- We need to talk to the customer and domain experts to identify the problem to be solved(“elicit the requirements”)
- Some problems are impossible to solve, others are realizable only after they are changed or reinterpreted
- Software engineering is about dealing with change and complexity while trying to solve the problem
- Software engineering uses methods such as divide&conquer and tools (CASE) to solve problems.

Backup Slides