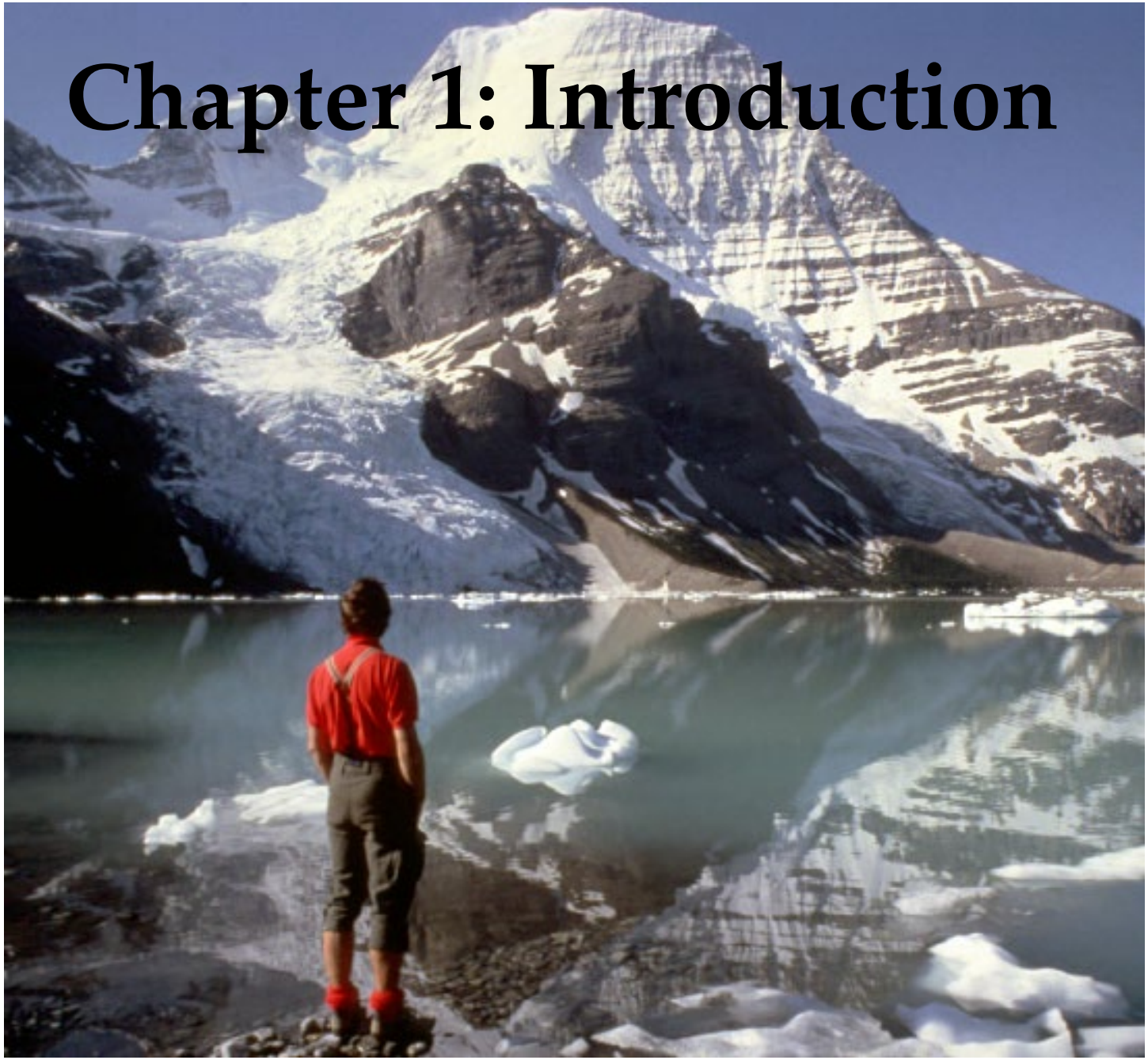
Object-Oriented Software Engineering
Using UML, Patterns, and Java

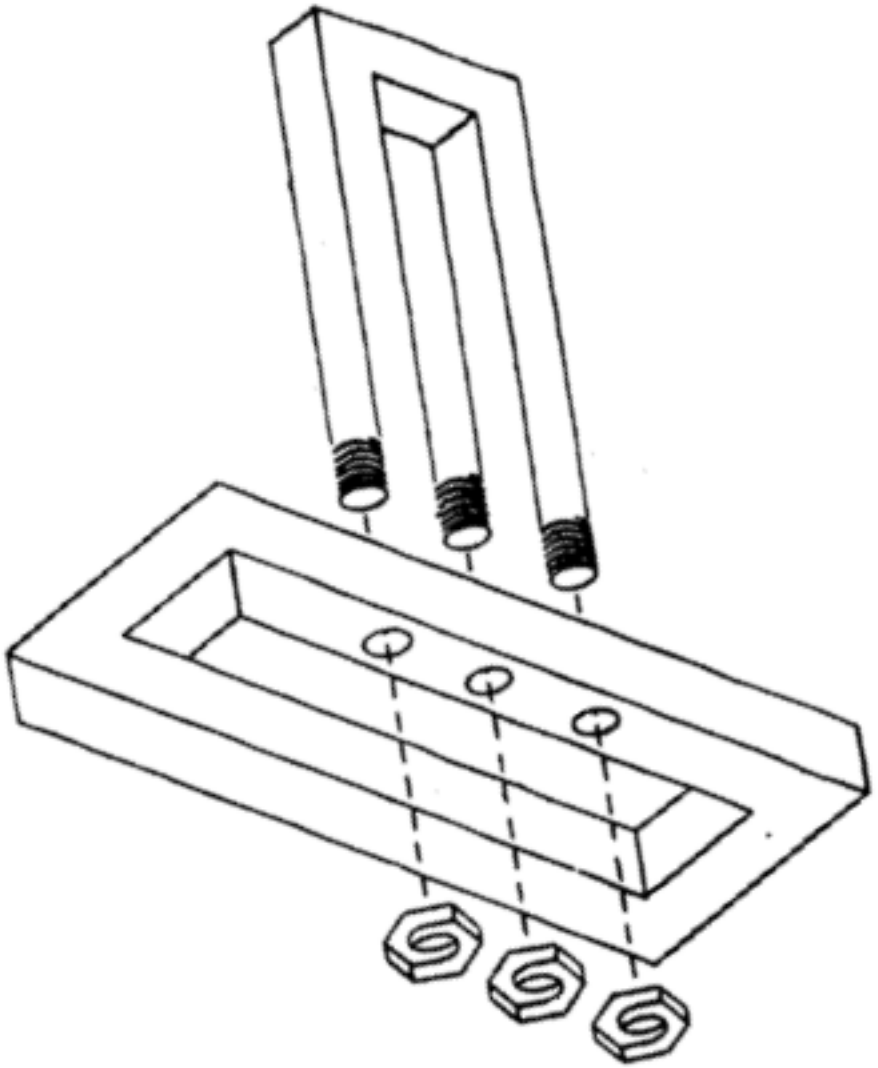
Chapter 1: Introduction



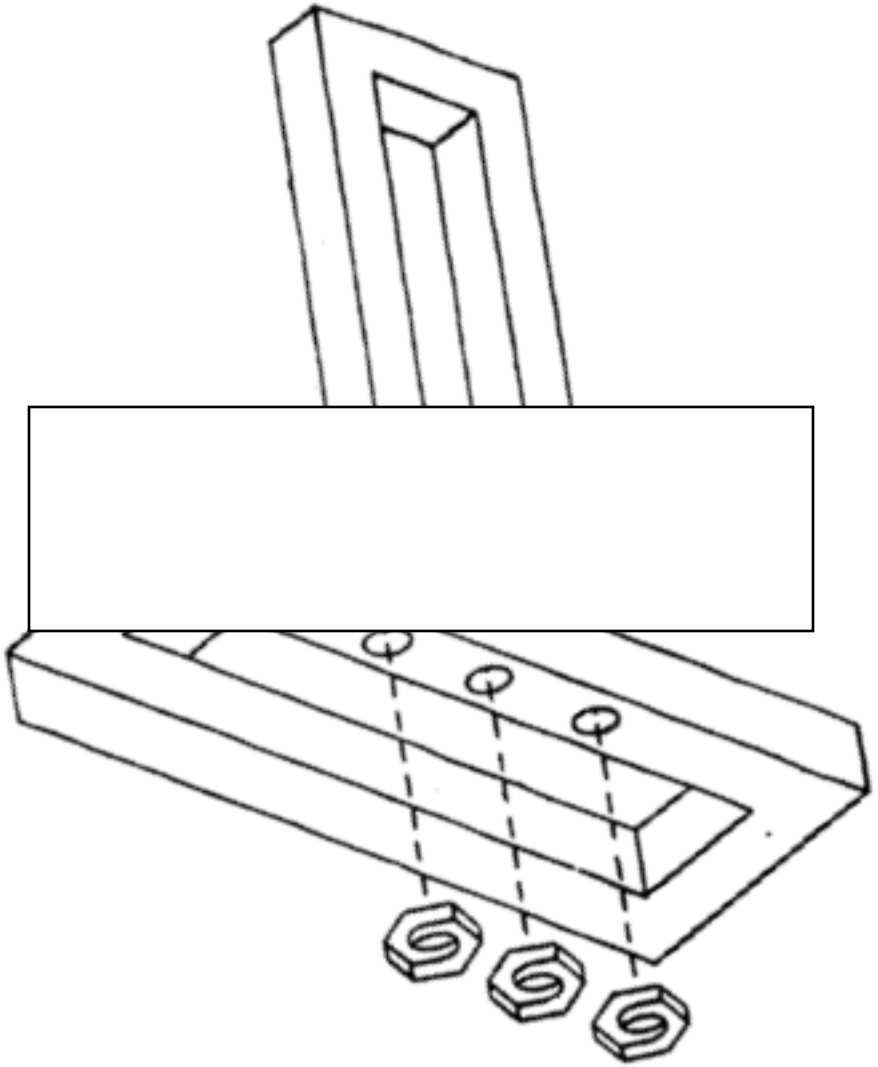
Outline of Today's Lecture

- The development challenge
- Dealing with change
- Concepts: Abstraction, Modeling, Hierarchy
- Methodologies
- Organizational issues

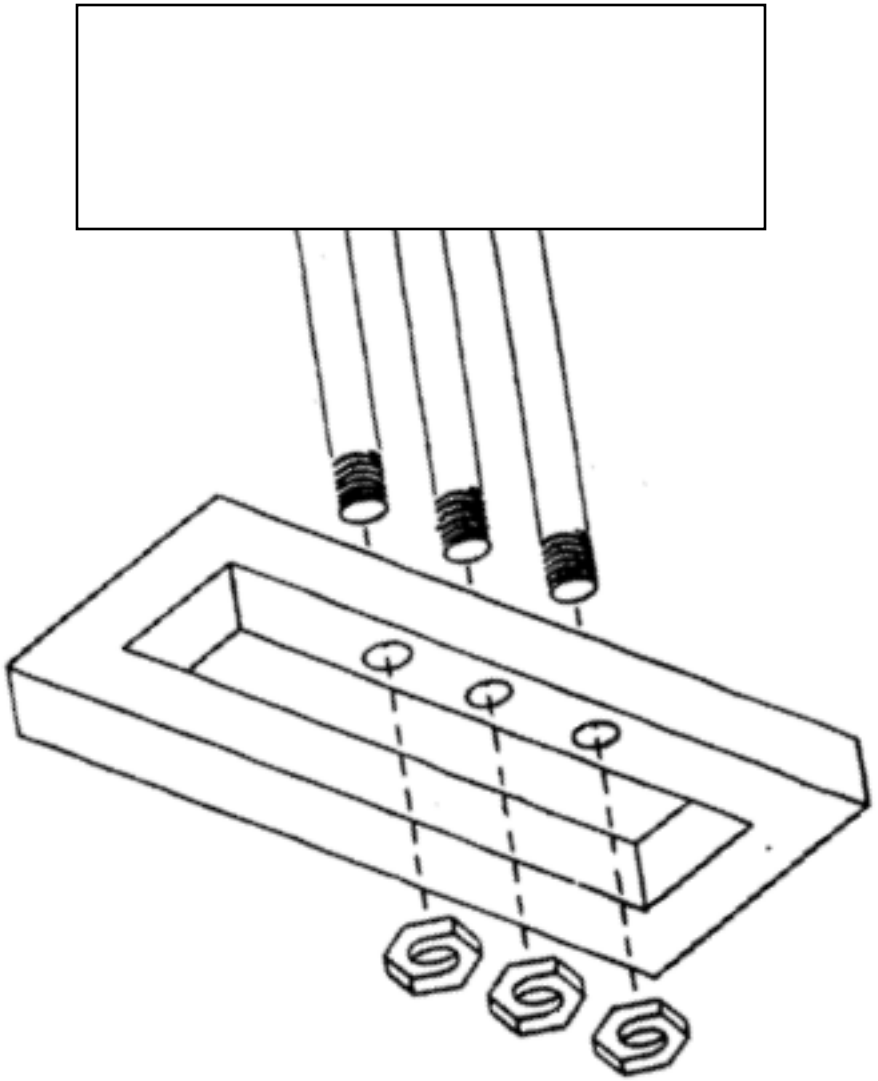
Can you develop this system?



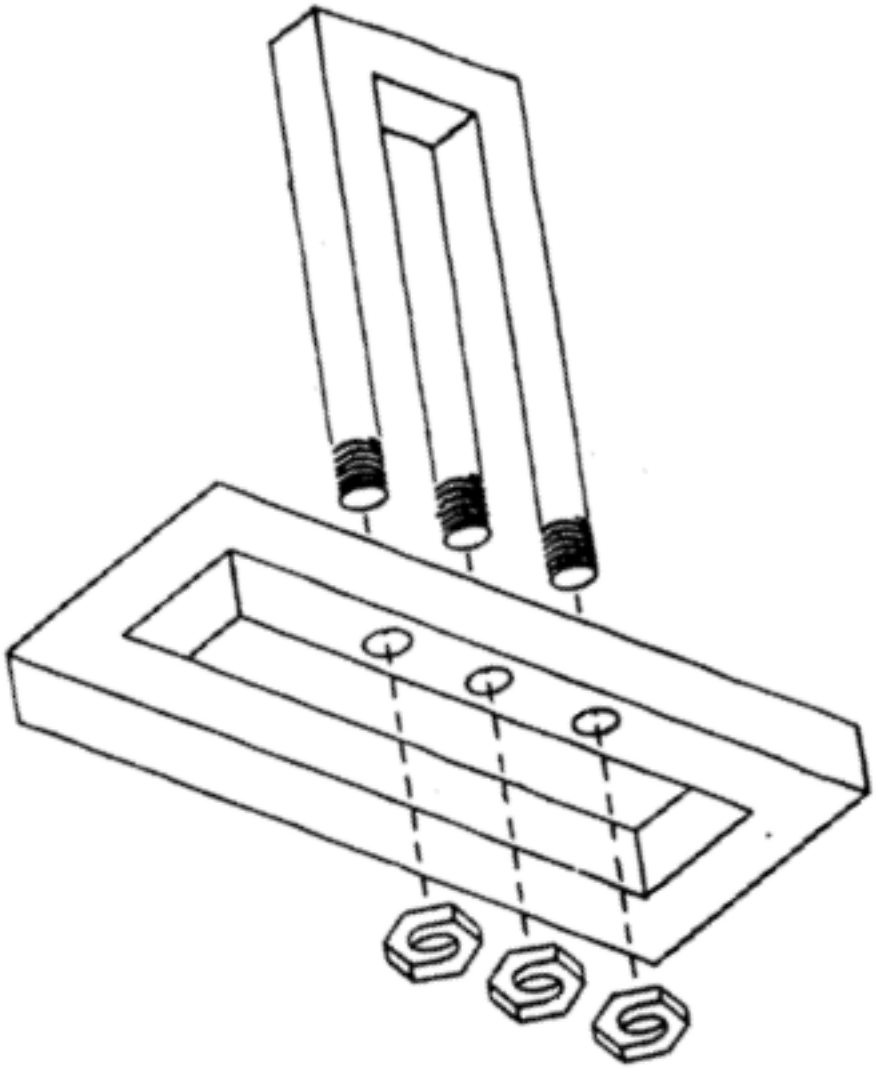
Can you develop this system?



Can you develop this system?



Can you develop this system?



The impossible Fork

Why is software development difficult?

- The problem is usually ambiguous
- The problem domain (also called application domain) is too large
- The solution domain is complex
- The development process is difficult to manage
- The requirements are usually unclear and changing when they become clearer
- Software offers extreme flexibility
- Eventually the end user must be satisfied.

Software Development is more than just Writing Code

- It is problem solving
  - Understanding a 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 knowledge management
  - Elicitation, analysis, design, validation of the system and the solution process
- It is rationale management
  - Making the design and development decisions explicit to all stakeholders involved.

"Programming is by far the hardest intellectual task human beings have ever tried to do."

Understanding the Professional Programmer
G.M. Weinberg

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 specific technique/method.
  - IDE = Integrated Development Environment
  - CASE = Computer Aided Software Engineering

Problems and Prospects

- Software Engineering (SE) is about the creation of large pieces of software.
- Equally, SE is about imagination and creativity.
- There is no single method for doing it.
- Instead, a whole variety of approaches.
- This diversity is one of the delights of SE.

Cruel Reality

Unreliable

Limited portability

Not always fast enough

Expensive to produce

Frequently delivered late

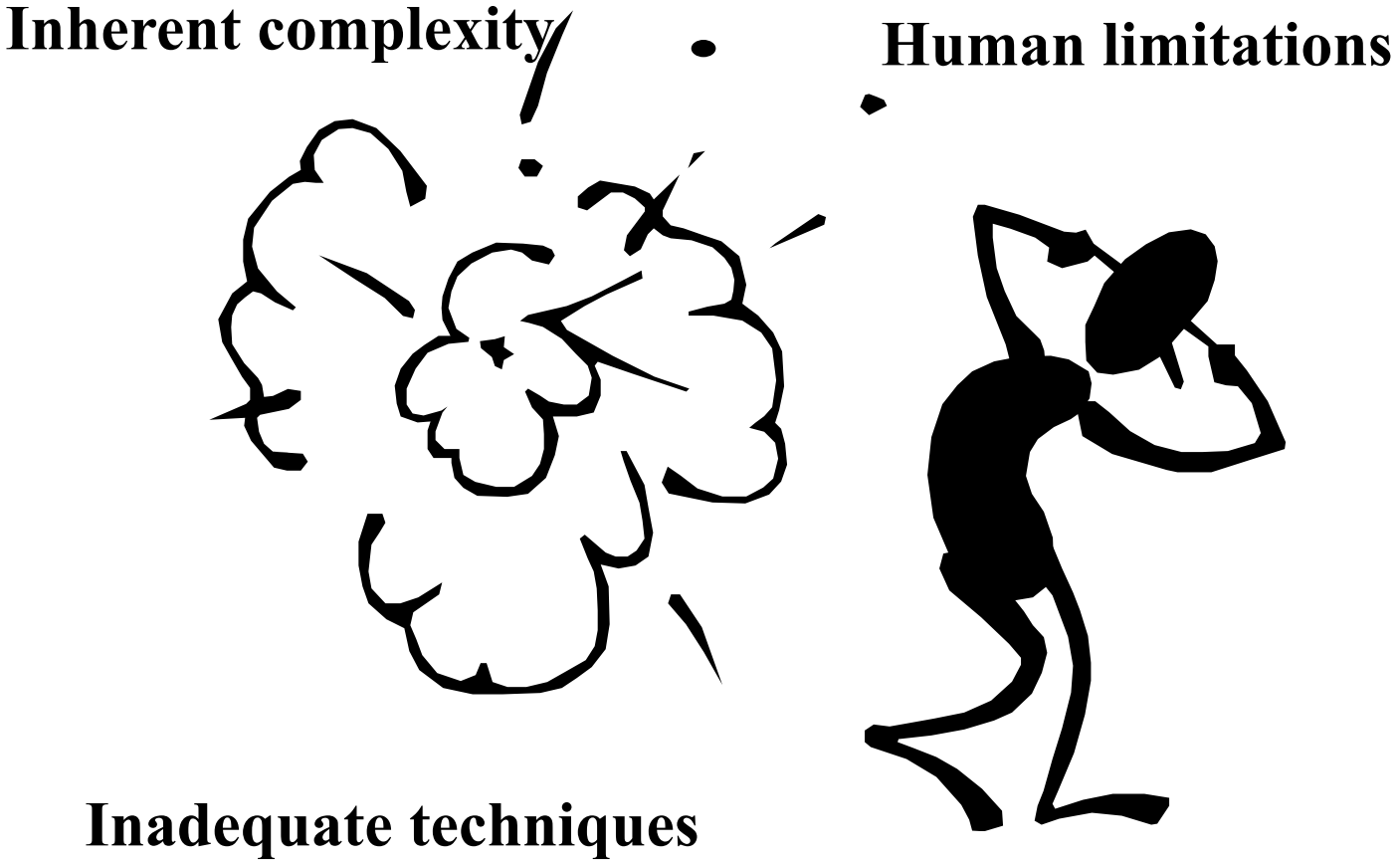
Difficult to maintain and evolve

Often fails to meet user want

Reasons for Software Failures

- Unrealistic or unarticulated project goals
- Poor project management
- Inaccurate estimates of needed resources
- Badly defined system requirements
- Poor reporting of the project's status
- Unmanaged risks
- Poor communication among customers, developers, and users
- Inability to handle the project's complexity
- Poor software design methodology
- Wrong or inefficient set of development tools
- Inadequate test coverage
- Inappropriate (or lack of) software process.

Software Crisis



Software Engineering: A Working 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

Software Engineering

- Replace ad hoc methods by a disciplined process
- Find out what the users really want
- New methodologies, approaches and tools
- Effective management
- Emphasis on carrying out software development systematically from requirements to maintenance.

Computer Science vs. Engineering

- **Computer Scientist**
  - Assumes techniques and tools have to be developed.
  - Proves theorems about algorithms, designs languages, defines knowledge representation schemes
- **Engineer**
  - Develops a solution for a problem formulated by a client
  - Uses computers & languages, techniques and tools
- **Software Engineer**
  - Works in multiple application domains
  - Has only 3 months...
  - …while changes occurs in the problem formulation (requirements) and also in the available technology.

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 large structure

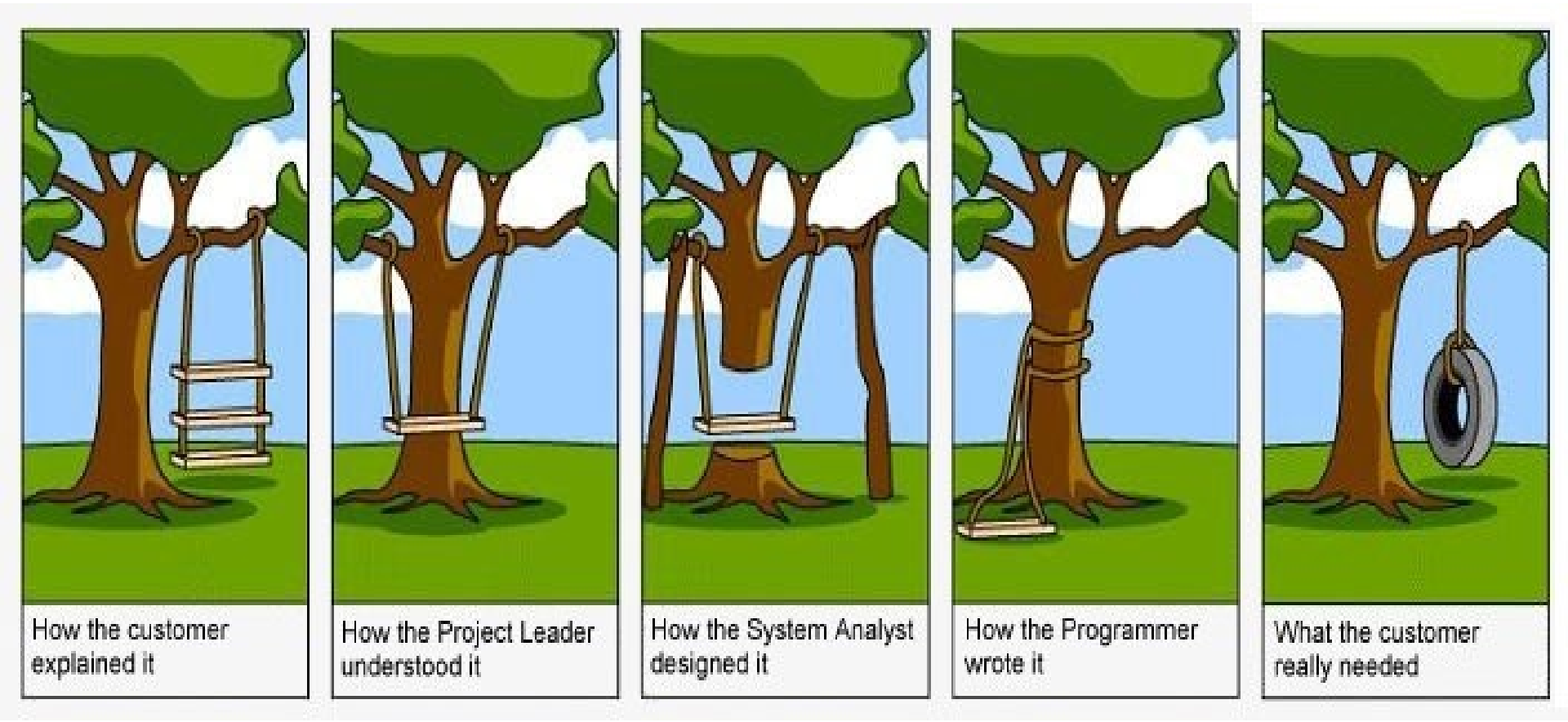
For problem solving we use techniques, methodologies and tools

Meeting Users' Needs

- Software must do what its users want; it seems self-evident, but…
- First step should be: find out user needs
- Called requirements analysis or requirements engineering
- A Software Requirements Specification (SRS) is written and approved by the stakeholders.

- Should be carried out with care, but…(Microsoft's good enough software).

Why is Software Requirements Important?



Give Products to Customers Early

- No matter how hard you try to learn user's needs during the requirements phase, the most effective way to determine real needs is to give users a product and let them play with it.

- Involve the customer early in the software life cycle.

Determine the Problem before Writing the Requirements

- When faced with what they believe is a problem, most engineers rush to offer a solution.
- Before you try to solve a problem, be sure to explore all the alternatives and don't be blinded by the "obvious" solution.

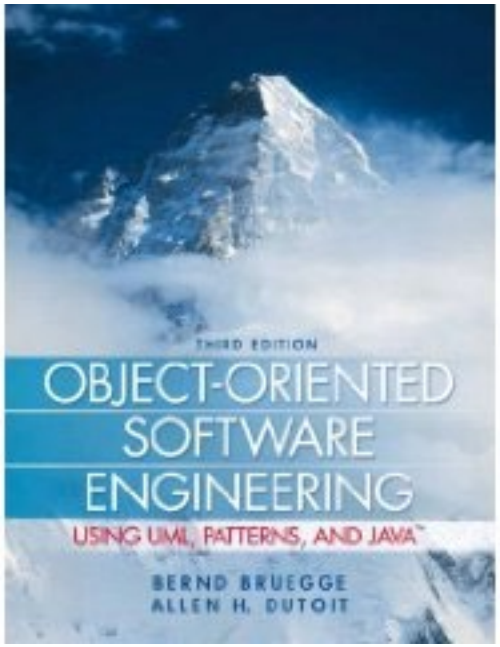
Famous Bugs – As We Speak

- Recent software bugs – Visit
- http://catless.ncl.ac.uk/Risks/

Visit the Course Portal

- Course Portal on Canvas:
  - The lecture slides will be posted in PDF format after the lecture is given
- "Labs":
  - Separate time will be set up for the exercises, assignments and project
- What happens if I don't attend lectures?
  - You are on your own.

Book from where these slides are based on:



Bernd Bruegge, Allen H. Dutoit

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

Publisher: **Prentice Hall**, 2010;

ISBN-10: 0136061257

ISBN-13: 978-0136061250

Readings for this Week-01

- Basic info about software engineering.
- Next week: software processes
- Later on: Unified Modeling Language (UML)
- Additional readings may be added during each lecture.

THINK!

- Re-examine your personal goal when developing software
- The conflicts of software engineering
- See Prezi presentations on my website
- https://www.eng.uwo.ca/Electrical/faculty/capretz_l/teaching.html

Exercise-1: Due in a week, upload to Canvas

Survey software tools and IDEs.

- Proprietary softwares
- Lots of open source software
- Lots of Free software

- For instance, this site contains a lot of software tools for testing software:
- www.softwareqatest.com/index.html