Clarification

- There is **NO** "textbook"
- The "lectures notes" are **Custom Courseware**
- SE 3MO4 (this course) and SE 3K04 share the same content but different instructors, lectures, slides and exams
- It so happens that the **Custom Courseware** only says SE 3K04 in its title.
- **Only** the *slides* will be posted on the web

What is "Software Engineering"

Engineering

- Engineering is a discipline
- **Goal**: Creation of a Product
- The Product is specified, documented, and it is possible to verify that it performs as it is supposed to
- An engineer is responsible for his/her product (Not a disclaimer!)

"Engineered" Software Products

- Pure software products
  - COTS: Microsoft Word, Matlab, PSpice
  - Banking system, MUGSI
- Software as part of others products
  (Car, Airplane, Nuclear Power Plant, X-ray-machine)
- Software used in the design of products
  - program computing the beams of a bridge

Mission and Safety Critical software

Definition:

- failure causes harm to life or environment
- financial disaster (recalls)

Who of you ever wrote a program larger than N lines which worked perfectly the first time?

How can we trust software consisting of 1'000'000 lines of code or more?

Goals of this course:

Learn how to develop (large) software products that

- are reliable
- really do what they are supposed to
- can be verified
- can be developed by a team
- can be easily maintained

Engineering Principles

- Accept individual responsibility
  - Social
  - Professional
  - Ethical
  - Environmental
- Solve the Real Problem
- Be honest about the capabilities
- Produce Reviewable Designs (Documentation)
- Specify and document your software
- State the limitations of your software

Software Facts

- More than half the cost of software is spent in maintenance
  - Bug Fixes
  - Change of Functionality
  - Extension of Functionality
- Software is used over a long time: "Software Aging" (Y2K problem, DOS/640K)
- Software is used for things it was not designed for
- More and more hardware is replaced by software; (A cellular phone has about 700k of software)
- Many crucial aspects of our life is controlled by software (ABS brakes, telephone, power, bank, medical instruments)

Software and Industry

- Software is often developed in an naive, unprofessional way
  - Year 2000 problem
  - "There are so many trees that we do not see the forest"
- Software is not documented
  - Knowledge is lost when a developer leaves
  - Maintaining the software is very difficult

Software and Industry

- There are no requirements specifications
  - Communication problems between programmer and client
  - "Is the software finished?"
- No formal or semi-formal specification
  - "How to test it?"
  - "Is is a bug or a feature?"
  - Why did this happen ? Should it happen? How can I prevent this from happening?