Welcome!

• Software Engineering is about **processes** to create software.

• This is a small (5 cr) course:
  – we study several processes, but
  – we do not execute any of them.

About the course

• Literature: Sommerville, 9th ed. *Buy and start reading - today!*
  • [http://www.it.uu.se/edu/course/homepage/pvt/vt13](http://www.it.uu.se/edu/course/homepage/pvt/vt13)
    – schedule, detailed reading instructions
      • lectures
      • guest lectures
      • seminar
    – examination

About the examination

• written exam (4 cr)
  • notes and selected parts of the book
  • grades U, 3, 4, 5
• assignment (1 cr)
  1. Select a SE article
  2. Hand in a summary + own opinion
  3. Give feedback to another student
  For details see the course homepage

Verbal skills (reading, writing) essential!

About the book

Part 1: Introduction to SE - mostly covered
Part 2: Dependability and Security - mostly covered
Part 3: Advanced SE - mostly *not* covered
Part 4: Management - mostly covered

This course is about **large** projects:

- 10 - 40 - … - 1000 people
- 3 - 12 - … - 100 months

Why do projects get large?
In a garage in Silicon Valley:

Program
3 people
6 months

x 3

System
- platforms
- interfaces, API
- customized

Product quality
Structure, names, encapsulation
Testable, tested, test suits
Maintainable - tech. doc.
User documentation

System Product
3 people, 54 months?
27 people, 6 months?
9 people, 18 months

SE is Engineering

- Making things that work - practical
- Use of models, standard designs, methods, tools
- Constraints: time, money, organisation
- Managing people, communication

...but not always like Engineering

- Every project is mostly new
- Software is "invisible",
  perceived to be adaptable
  (rewrite code vs. rebuild a bridge)
- lacks physical boundaries - gets complex
- Usability - "getting it right"
- Legacy systems

Process

- Process
  - what really happens … too messy
- Process model
  - abstraction, common themes
- Method (or: Process)
  - what should happen
  Every company has one! (ADM, PROPS)
Product Quality

- Acceptable: usable, learnable, compatible
- Efficient: response time, memory use
- Dependable: safe, reliable, secure
- Maintainable: documented, structured

[Fig. 1.2, 24.2]

Process (method) Quality

- People actually follow it:
  - acceptable, usable, learnable
- It delivers:
  - efficient, in time, acceptable product quality
- Manageable
  - visible, robust to problems
- Supportable
  - documented, structured

The ingredients

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Design</th>
<th>Implementation</th>
<th>Testing</th>
<th>Operation, Maintenance</th>
</tr>
</thead>
</table>

Some related courses

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Design</th>
<th>Implementation</th>
<th>Testing</th>
<th>Operation, Maintenance</th>
</tr>
</thead>
</table>

The waterfall model

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Design</th>
<th>Implementation</th>
<th>Testing</th>
<th>Operation, Maintenance</th>
</tr>
</thead>
</table>
The V-model

Requirements → System tests → Operation, Maintenance

Test Plans → Integration tests → Implementation → Unit tests

Validation

- Will the product satisfy the customer needs?
- Are we building the right product?

Verification

- Do we satisfy the requirements?
- Are we building the product right?

The V-model (Sommerville)

Prevas project model (detailed)

Outline of the course

- The V-model as a roadmap
- Dependable systems
- Project management
- Ethics

Figure 2.11: Boehm’s spiral model of the software process
Project start

• Custom made
• Generic (COTS)

• Extremely customizable systems
  (ERP systems, like SAP)

Commercial
Off The Shelf

Feasibility study

Many IT-projects that fail, should not have started

• One System to integrate them all ...
• Organisation that is the problem, not IT
• Political action (we need to do something ...)
• We can make it work ...

Describe the environment with the new IT-system
Ethnographical study [4,5,5]