Chapter 1: Introduction

Objectives of the Lectures

• Appreciate the Fundamentals of Software Engineering:
  • Methodologies
  • Process models
  • Description and modeling techniques
  • System analysis - Requirements engineering
  • System design
  • Implementation: Principles of system development

Assumptions for this Class

• Assumption:
  • You have taken Module 006 EIST: Introduction to Software Engineering or a similar course
  • You have already experience in at least one analysis and design technique

• 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.

Times and Locations

• Main lecture: HS 2
  • Tuesdays 16:00 – 18:00
  • Fridays 9:00 - 10:00

• Exercises: Thursday 8:00-10:00
  • Registration starts today
  • Registration ends Thursday
  • Exercise sessions start on Thursday, Oct 16

• Written Exams:
  • Mid-term
  • Final

Grading Criteria

The final grade is the weighted average of the mid term (30%) and final grades (70%).

• Participation in the exercises is required (admission requirement for the final exam)
• Information about the exercises will be made available on the exercise portal
• Hours per week: 3 hours (lecture) + 2 hours (exercises)
• ECTS Credits: 6.0.

Focus: Acquire Technical Knowledge

• Different methodologies ("philosophies") to model and develop software systems
• Different modeling notations
• Different modeling methods
• Different software lifecycle models (empirical control models, defined control models)
• Different testing techniques (e.g., vertical testing, horizontal testing)
• Rationale Management
• Release and Configuration Management
Acquire Managerial Knowledge

- Learn the basics of software project management
- Understand how to manage with a software lifecycle
- Be able to capture software development knowledge (Rationale Management)
- Manage change: Configuration Management
- Learn the basic methodologies
  - Traditional software development
  - Agile methods.

Outline of Today’s Lecture

- The development challenge
- Dealing with change
- Concepts: Abstraction, Modeling, Hierarchy
- Methodologies
- Organizational issues
  - Lecture schedule
  - Exercise schedule
  - Associated Project

Can you develop this system?

The impossible Fork
Physical Model of the impossible Fork (Shigeo Fukuda)

See http://illusionworks.com/mod/movies/fukuda/DisappearingColumn.mov

Physical Model of the impossible Fork (Shigeo Fukuda)

Additional material can be found on http://illusionworks.com/mod/movies/fukuda/ Images may be subject to copyright

Why is software development difficult?

- The problem domain (also called application domain) is difficult
- The solution domain is difficult
- The development process is difficult to manage
- Software offers extreme flexibility
- Software is a discrete system
  - Continuous systems have no hidden surprises
  (Parnas)

Software Engineering is more than writing Code

- Problem solving
  - Creating a solution
  - Engineering a system based on the solution
  - Modeling
  - Knowledge acquisition
  - Rationale management

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
  - CASE = Computer Aided Software Engineering

Computer Science vs. Engineering

- Computer Scientist
  - Assumes techniques and tools have to be developed.
  - Proves theorems about algorithms, designs languages, defines knowledge representation schemes
  - Has infinite time...
- 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 occur in the problem formulation (requirements) and also in the available
### 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

- **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.

---

### Course Outline

**Dealing with Complexity**
- Notations (UML, OCL)
- Requirements Engineering, Analysis and Design (DOSE, SA/SD, scenario-based design, formal specifications)
- Testing (vertical and horizontal testing)

**Dealing with Change**
- Rationale Management
- Knowledge Management
- Release Management
- Big Bang vs Continuous Integration
- Software Life Cycle (Linear models, Iterative models, Activity-oriented vs Entity-based views)

---

### Tentative Lecture Schedule

<table>
<thead>
<tr>
<th>Block 1 (Oct 14 – Oct 17)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction, Methodologies</td>
</tr>
<tr>
<td>The UML notation</td>
</tr>
<tr>
<td>Build and Release Management</td>
</tr>
<tr>
<td>Testing, Programming Contest</td>
</tr>
<tr>
<td>Software Lifecycle Models</td>
</tr>
<tr>
<td>Midterm Exam: Dec 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 2 (Oct 18 – Nov 14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notations (UML, OCL)</td>
</tr>
<tr>
<td>Requirements Engineering, Analysis and Design (DOSE, SA/SD, scenario-based design, formal specifications)</td>
</tr>
<tr>
<td>Testing (vertical and horizontal testing)</td>
</tr>
<tr>
<td>Rationale Management</td>
</tr>
<tr>
<td>Knowledge Management</td>
</tr>
<tr>
<td>Big Bang vs Continuous Integration</td>
</tr>
<tr>
<td>Software Life Cycle (Linear models, Iterative models, Activity-oriented vs Entity-based views)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 3 (Nov 18 – Nov 21)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build and Release Management</td>
</tr>
<tr>
<td>Testing, Programming Contest</td>
</tr>
<tr>
<td>Software Lifecycle Models</td>
</tr>
<tr>
<td>Midterm Exam: Dec 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 4 (Nov 25 – Nov 28)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale Management</td>
</tr>
<tr>
<td>Knowledge Management</td>
</tr>
<tr>
<td>Big Bang vs Continuous Integration</td>
</tr>
<tr>
<td>Software Life Cycle (Linear models, Iterative models, Activity-oriented vs Entity-based views)</td>
</tr>
<tr>
<td>Software Lifecycle Models</td>
</tr>
<tr>
<td>Midterm Exam: Dec 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 5 (Dec 1 – Dec 13)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build and Release Management</td>
</tr>
<tr>
<td>Testing, Programming Contest</td>
</tr>
<tr>
<td>Software Lifecycle Models</td>
</tr>
<tr>
<td>Midterm Exam: Dec 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 6 (Dec 16 – Jan 9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale Management</td>
</tr>
<tr>
<td>Knowledge Management</td>
</tr>
<tr>
<td>Big Bang vs Continuous Integration</td>
</tr>
<tr>
<td>Software Life Cycle (Linear models, Iterative models, Activity-oriented vs Entity-based views)</td>
</tr>
<tr>
<td>Software Lifecycle Models</td>
</tr>
<tr>
<td>Midterm Exam: Dec 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 7 (Jan 10 – Feb 9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale Management</td>
</tr>
<tr>
<td>Knowledge Management</td>
</tr>
<tr>
<td>Big Bang vs Continuous Integration</td>
</tr>
<tr>
<td>Software Life Cycle (Linear models, Iterative models, Activity-oriented vs Entity-based views)</td>
</tr>
<tr>
<td>Software Lifecycle Models</td>
</tr>
<tr>
<td>Midterm Exam: Dec 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block 8 (January and February)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific methodologies: XP, Scrum, Rugby, Royce</td>
</tr>
<tr>
<td>Plus invited lectures from industry</td>
</tr>
<tr>
<td>Final Exam: Feb 9</td>
</tr>
</tbody>
</table>

---

### Exercises

- The exercises are organized in two groups
- Each group has one exercise session (2 hours) per week: Thursday 8:00–10:00
- Registration, attendance in the exercise sessions and attempting/finishing the homeworks is mandatory.

---

### Textbook

**Bernd Brügge, 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  

- Additional readings will be added during each lecture.
Questions?

- Lecture Portal:
  - The lecture slides will be posted in PDF format after the lecture is given
- Exercise Portal:
  - Separate home page will be set up for the exercise materials
- What happens if I don’t participate in the exercises?

Play the movie
http://www.youtube.com/watch?v=_VFS8zRo0pc