Lecture 1: Introduction to Software Engineering and Software Process

Quality Methodologies for Software (INSE 6250/4-UU)
Winter 2009

Overview

- Course Outline
- Software Engineering
- Software Process Models
- Project

Course Outline

- Instructor: Dr. J. Bentahar
- Office: EV7.630
- Lectures: Tuesday, 20h30 – 23h00
- Office Hours: Wednesday, 14h00 – 16h00 or by appointment
- Phone: 848-2424 ext. 5382
- E-Mail: bentahar@ciise.concordia.ca

Course Web:
http://www.ciise.concordia.ca/~bentahar/inse6250.html
- Lecture notes
- Assignment
- Useful links
- Useful information
**INSE 6250/4-UU**

- Quality Methodologies for Software:
  - Software engineering (processes, life cycles, and specifications)
  - Quality methodologies for these processes, life cycles, and specifications

- Objectives:
  - To discover and learn various concepts and techniques related to software engineering and quality methodologies
  - To learn to apply these techniques
  - To develop critical thinking skills

**Textbooks**

1) *Quality Software Project Management, 2002*

- Process overview, software developments life cycles, developing the software requirements specification, software engineering, software metrics, and validation and verification
Textbooks

2) *The B-Book: Assigning Programs to Meanings*, 1996

- This book is a reference in the B method for software specification. It covers a comprehensive breadth of this method and its applications.

Textbooks

3) *Understanding Formal Methods*

- Overview of formal methods for software development, logical tools, and set-theoretic specifications (Z, VDM, and B)

Textbooks


- Verification and Validation
- Available from the course web site

Requirements and Grading

- One individual/group assignment
  - 15% (Two parts: Technical part + Synthesis part)
- One in-class midterm exam (closed book)
  - 25%
- One in-class final exam (closed book)
  - 30%
- One team project (2~3 members, presentation + report)
  - 15% + 15% = 30%
**Important dates**

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project proposal</td>
<td>February 03, 2009</td>
</tr>
<tr>
<td>Assignment</td>
<td>February 10, 2009</td>
</tr>
<tr>
<td>Midterm exam</td>
<td>March 03, 2009</td>
</tr>
<tr>
<td>Project presentation</td>
<td>March 31, 2009</td>
</tr>
<tr>
<td>Final exam</td>
<td>TBS</td>
</tr>
<tr>
<td>Project report</td>
<td>April 10, 2009</td>
</tr>
</tbody>
</table>

**Project**

- Project proposal
  - Deadline: February 03, 2009
- Team members
- Topic and title
- Abstract
- Main references

**Questions**

- Please describe briefly about yourself:
  - Academic background
  - Background in **programming** and **software engineering**
  - Industrial experience
  - Research domains you are interested in

**Overview**

- Course Outline
- Software Engineering
- Software Process Models
- Project
Software Engineering

- The application of engineering to software
- Field of computer science dealing with software systems
  - Large and complex
  - Built by teams
  - Exist in many versions
  - Undergo changes

Definitions

- Application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software (IEEE 1990)
- Multi-person construction of multi-version software (Parnas 1978)

Importance of Software Engineering

- The economies of all developed nations are dependent on software
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development

Role of Software Engineering in System Design

- Software engineering is part of larger projects
- Embedded
  - Software requirements to be balanced against others
  - e.g., telephone switching systems
- Compromise: what should be done in software and what should be done in hardware
History

- The field of software engineering was born in 1968 in response to chronic failures of large software projects to meet schedule and budget constraints
  - Recognition of "the software crisis"
- Term became popular after NATO Conference in Garmisch Partenkirchen (Germany), 1968

Software Costs

- Software costs often dominate computer system costs
- The costs of software on a PC are often greater than the hardware cost
- Software costs more to maintain than it does to develop
- Systems with a long life, maintenance costs may be several times development costs

Software Engineering Aspects

- Design
- Modeling
- Programming
- Customer Request
- Testing
- Verification
- Management
- Quality Software

Role of Software Engineer

- Programming skill not enough
- Software engineering involves "programming-in-the –large"
  - Understand requirements and write specifications
  - Derive models and reason about them
  - Operate at various abstraction levels
- Member of a team
  - Communication skills
  - Management skills
Relationships with Other Computer Science Disciplines

- Programming Languages
- Operating Systems
- Networking
- Software Engineering
- Data Bases
- Artificial Intelligence
- Theory

Synergetic Relationships

Systems Engineering V.S. Software Engineering

- Systems engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering
- Software engineering is part of this process concerned with developing the software infrastructure, control, applications and databases in the system
- System engineers are involved in system specification, architectural design, integration and deployment

Overview

- ✔ Course Outline
- ✔ Software Engineering
- ✔ Software Process Models
- ✔ Project
Software Process Model

- Attempt to organize the software life cycle by
  - Defining activities involved in software production
  - Order of activities and their relationships
- Goals of a software process
  - Standardization, predictability, productivity, high product quality, ability to plan time and budget requirements

The Earliest Approach

- Write code
- Fix it to eliminate any errors that have been detected, to enhance existing functionality, or to add new features
- Source of difficulties and deficiencies
  - Impossible to predict
  - Impossible to manage
Software Process Models

- Waterfall
- Vee
- Spiral
- Others

Waterfall Model

- Introduced by Royce in 1970
- Each phase is carried out to completion in sequence until the product is delivered
- This is not possible in all cases

Vee Model

- The model starts with user needs and ends with a user validated system
- Left side: the system architecture
- Right side: Integration and verification
- Middle side: testing

Spiral Model
Spiral Model

- Developed by Boem from 1969 to 1986
- Risk driven approach
- Adaptation of the waterfall model (the use of prototypes)
- Iterative application: each time a different type of prototype is developed

Evolutionary Models

- Many variants available
- Product development evolves through increments
  - evolutionary prototype
- Evolutionary process model (B. Boehm, 1988)
  "model whose stages consist of expanding increments of an operational software product, with the direction of evolution being determined by operational experience"

Transformation Model

- Guided by formal methods
- Specifications transformed into implementations via transformations
- Still a theoretical reference model
What are the Costs of Software Engineering?

- Roughly 60% of costs are development costs, 40% are testing costs.
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- Distribution of costs depends on the development model that is used.

Activity Cost Distribution

- **Waterfall model**
  - Specification: 50%
  - Design: 25%
  - Development: 25%
  - Integration and testing: 100%

- **Iterative development**
  - Specification: 25%
  - Iterative development: 50%
  - System testing: 75%
  - System development: 100%

- **Component-based software engineering**
  - Specification: 25%
  - Development: 50%
  - Integration and testing: 75%
  - System development: 100%

- **Long-lifetime systems**
  - System development: 100%
  - System evolution: 40%

Specification

- A broad term that means *definition*.
- Used at different stages of software development for different purposes.
- Generally, a statement of agreement (*contract*) between:
  - Producer and consumer of a service.
  - Implementer and user.
- All desirable qualities must be specified.
### Specification

- **Types of specifications**
  - **Operational**
    - Data Flow Diagrams
    - (Some) UML diagrams
    - Finite State Machines
    - Petri Nets
  - **Descriptive**
    - (Some) UML diagrams
    - Entity Relationship Diagrams
    - Logic-based notations
    - Algebraic notations

### Specification

- **Languages for modular specifications**
  - Statecharts
  - Activity Diagram
  - Logic-based Languages: PLTL, CTL, CTL*
  - Z
  - B
  - VDM: Vienna Development Method
  - SDL: Specification & Description language

### Software Evolution

- Existing software must evolve because requirements change
- Re-engineering
  - Process through which an existing system undergoes an alteration, to be reconstituted in a new form
- Reverse engineering
  - From an existing system to its documentation, which may not exist, or may be incomplete
  - Includes program understanding

### Overview

- ✓ Course Outline
- ✓ Software Engineering
- ✓ Software Process Models
- ✓ Project
Example of topics:

- Software engineering for Web services and semantic Web
- Agent-based software engineering
- Service-oriented software engineering
- Concert cases for model checkers: Spin and Promela, SMV, NuSMV, Uppaal, etc…

Development Using any kind of technology: Java, Jack, Jade, Prolog, model checker language, etc.

Key challenges facing software engineering: heterogeneity and trust

- Heterogeneity: developing techniques for building software that can cope with heterogeneous platforms and execution environments
- Trust: developing techniques that demonstrate how trust is managed
  - Agent-based systems