Course Information
EECS 393/493: Software Engineering
Falls 2017

Instructor: Andy Podgurski
Office: Olin 510
Office hours: TBA
Phone: 368-6884
Email: podgurski@case.edu
TAs: TBA

Course Description
Software engineering is both a synonym for the systematic use of well-founded software development practices and the name of the area of computer science and engineering that deals with the specification, design, implementation, validation, and maintenance of complex software systems and applications. This course covers the main elements of software engineering methodology. Undergraduate students usually apply them in the setting of a significant team project. Alternatively, they can participate in a research project, with instructor approval. Graduate students are required to investigate a research problem in software engineering.

Objectives
- Understanding of the software development lifecycle
- Understanding of the issues and challenges associated with each phase of software development
- Familiarity with the basic software engineering techniques and tools for each phase of software development
- For undergraduates, experience working as part of a team on a substantial software development project
- For graduate students, experience investigating a software engineering research issue in depth
- Cultivation of critical thinking skills needed by professional software engineers and software engineering researchers

Prerequisites
- Intermediate to advanced programming skills in at least one modern programming language, such as Java, C++, C#, or Python
- EECS 233 with a grade of C or higher.

Textbook: None required.

Canvas: Slides and other materials will be posted on Canvas. However, these are *not* a substitute for attending class. You are responsible for what is discussed in class,

whether or not it appears in the slides. Almost every topic discussed in class is also discussed on various software engineering web sites, which you are encouraged to consult for more information and different perspectives.

Grading

Quizzes and homeworks (60%)
Project (40%)

Project

Undergraduate students are expected to complete a software development project, whose work products include: a software requirements specification, a design document, functional test-case descriptions, implementation code, unit tests, and a basic user's manual. Teams of two to five students may work together on a development project. Projects are proposed by each team and must be approved by the instructor. Scheduled demonstrations of work products and development practices (e.g. code, tests, version control) are required. Each team member must contribute significantly to each project work product or deliverable.

Undergraduates may optionally choose to do a research project on a topic suggested or approved by the instructor. The topic will generally involve implementation (e.g., developing a software analysis tool) and/or empirical evaluation (e.g., of a software testing or analysis technique).

Graduate students are expected to complete a research project, whose work products include: a written project proposal, including a literature survey; a modest *original* research contribution; a written progress report; a mid-semester demonstration to the instructor; and a final written project report and demonstration. The research contribution may take one of two forms: (1) a well-designed study clarifying an important issue in software engineering research, typically through implementation and comparative empirical evaluation of two or more existing or proposed techniques for a given software engineering problem; or (2) development and empirical evaluation of an innovative and potentially useful technique or tool. Teams of up to four students may work together on a research project. Projects are proposed by each team and must be approved by the instructor. Research reports should use the approved formats for either ACM or IEEE research conference papers. All material used from papers or other sources should be fully cited.

Detailed Syllabus (tentative)

Introduction (Week 1)
- What is Software Engineering?

The Software Development Process (Weeks 1 and 2)
- Lifecycle Models

- Waterfall Model
- Rapid Prototyping
- Incremental Development
- Agile Development
- Spiral Model

Software Requirements (Weeks 3-4)
- What are Software Requirements?
- Requirements Elicitation and Analysis
- Specifying Requirements
- Validating Requirements
- Maintaining Requirements

Software Design (Weeks 5-8)
- Design Dimensions and Principles
  - Managing Complexity
  - Designing for Change
  - Divide and Conquer
  - Encapsulation and Information Hiding
- Design Techniques
  - Stepwise Refinement
  - Object-oriented Design
  - Event-Driven Design
- Design Patterns
- Design Languages and Notations
  - UML
- Design Documentation
- Design Tools
- Design Inspections and Reviews

Validating Software (Weeks 9-12)
- Testing
  - Testing Phases
    - Unit Testing
    - Subsystem Testing
    - Integration Testing
    - System Testing
    - Acceptance Testing
    - Beta Testing
  - Synthetic Testing
    - Specification-based Testing
    - Code-based Testing
    - Fault-based Testing
    - Interaction Testing

- Field Testing
- Regression Testing
- Code Inspection
- Static Analysis Tools
- Statistical Reliability Estimation
- Post-deployment Validation

Product Delivery (Weeks 13-14)

- Configuration Management
- Continuous Integration
- Defect Tracking
- Deployment Pipeline
- Build and Deployment Scripting
- Deploying and Releasing Applications
- Delivery Ecosystem

Project Presentations (Week 15)