Course Syllabus : IPSE v2 - roy.canseco@ieee.org

CS 173-IPSE 2nd Semester 2017-18

Course Description

This is an elective course for discussing and experiencing Industrial Practices in Software Engineering (IPSE). There is emphasis on working on existing software code bases in a team context. A secondary emphasis is given to automated testing and continuous integration.

Prerequisite: 3rd year standing or Consent of Instructor (COI)

Credits: 3 units

Meeting Type: Lecture

Course Objectives

At the end of the course, the student should be able to:

1. Differentiate between the programming practices and emphases of Industry against those of Academic Institutions and discuss the conditions that promote such differences.

2. Cite at least three important reasons for having systematic code testing strategies in place, explain why Industry is partial to automated testing methods, and discuss possible reasons why Academe leans against requiring automated tests in school programming projects.

3. Experience being in a team that uses Test Driven Development (TDD) and sets up Continuous Integration (CI) for a software project and be able to outline steps in setting up CI and using TDD.

Course Requirements

Topic	Skill	Assessment	Percentage
Project Base Code	(code review)	Git repo submission	10 %
Feature Improvement	(team software development)	Feature Demonstration	10%

Topic	Skill	Assessment	Percentage
Test Cases	(analysis and design)	Midsem Presentation	20%
Automated Testing	(test environment setup)	Red to Green Demonstration	30%
Continuous Integration	(deployment and build pipeline)	FInal Presentation	30%

Expanded Course Outline

1. Project Code Base
    1. Students obtain existing code bases from other courses to use as the existing code base
    2. Nominated code bases are evaluated for possible use in the course according to a criteria 1. non-familiarity: most if not all of the students of the team should not have been a part of the creation of the code base 2. complexity: the code base should be complex enough to warrant a team (3-8) members.
2. Improvements
    1. A list of New Features for the code base is brain stormed
    2. The Prof. chooses several to make the degree of difficulty more or less fair among the teams
3. Tests
    1. Unit tests are formulated for all computing functions.
    2. Integration tests are formulated for all core features.
4. Test Automation
    1. A testing framework is chosen
    2. UI testing is explored
5. Continuous Integration
    1. A test suite of 30-100 tests is selected to be run at every build
    2. Development is continued with CI and the project health is reported along with CI metrics at every meeting after until Final project presentation.

References

- Kent Beck, 2002, Test-Driven Development by Example, Three Rivers Institute
- John Ferguson Smart, 2011, Jenkins the Definitive Guide - Continuous Integration for the Masses, O'Reilly Media, CA