Software Construction SWEN6301

Instructor: Ahmed Tamrawi

Fall 2018, Revision 2

E-mail: ahmedtamrawi@gmail.com Web: https://atamrawi.github.io/teaching/swen6301
Private Meetings: Available upon request Class: Saturdays 2:00 – 4:50pm (Masri 402)

This course syllabus should be interpreted as a contract of understanding between students, teaching assistants, and the instructor. By participating in the course you are agreeing to this understanding. Please review this document carefully and inform the instructor of any concerns. Revisions to this document will be made as needed then posted and announced in class.

Course Description

Software is everywhere and despite the rapid advances in software development practices and integrated development environments (IDEs), we still produce software that are buggy, late, over budget, and many fail to satisfy the needs of their users. The Software Construction course is about how to perform software development for delivering robust and resilient solutions which minimize the occurrence of the aforementioned issues. This course aims at providing the theories and techniques of effective and maintainable software development by using modern technologies and concepts, focusing on software construction.

The *primary goal* of this course is to help you to create higher-quality software that is robust, flexible, extensible, scalable, and maintainable. We focus on doing that by understanding what are common software development practices. The other main goal of this course is to provide experiences and knowledge that will help you develop as professional programmers and computing system designers. This includes: (1) experience working in a team, both as a leader and contributor, and (2) experience using tools commonly used by productive developers.

If you do not feel my goals for the course align well with your personal goals, but you need to take this course anyway to satisfy a degree requirement, you should meet with me to figure out a way to make this course useful for satisfying your personal goals.

Expected Background & Prerequisites

Students entering this course are expected to be comfortable reading, designing, and writing *Java* programs that involve code distributed over many modules. You should be comfortable learning how to use new programming language features and APIs by reading their documentation (or source code when no documentation is available), and not be surprised when solving programming assignments that require you to seek documentation beyond what was provided in class.

Required Materials

We will closely follow the textbooks from:

- Steve McConnell's *"Code Complete: A Practical Handbook of Software Construction,"* 2^{nd} Edition, (ISBN 978-0735619678).
- Robert C. Martin's *"Clean Code: A Handbook of Agile Software Craftsmanship,"* 1^{st} Edition (ISBN 978-0132350884).

In addition, we will have several readings from many other resources including:

- Hans-Petter Halvorsen's *"Software Development - A Practical Approach!'* (ISBN 978-82-691106-0-9).
- Joshua Bloch's *"Effective Java,"* 3^{rd} Edition (ISBN 978-0134685991).
- Edward Crookshanks's *"Practical Software Development Techniques: Tools and Techniques for Building Enterprise Software,"* 1^{st} Edition (ISBN 978-1484207291).

- Frank Tsui, Orlando Karam, and Barbara Bernal 's *"Essentials of Software Engineering,"* 4^{th} Edition (ISBN 978-1284106008).
- Ian Sommerville's *"Software Engineering,"* 10^{th} Edition (ISBN 978-0133943030).

Course Structure

Classroom Meetings

Based on the registrar's schedule, the lecture will last for 170 minutes. We will divide it into multiple parts upon class agreement.

Assignments

There will be 3-4 assignments through the course. Assignments will consist of two types of questions: *synthesis* and *conceptual*. Synthesis questions require you to analyze a program with respect to the software construction concepts and practices studied in class. Conceptual questions will reinforce important concepts and may only be practically achievable with a working software implementation. Partial credit is always available so do your best to complete as much as you can in each assignment.

Research Paper

Every student is expected to write a research paper on a topic of interest. A list of suggested topics will be posted. Students are free to choose other topics as long as the instructor approves it. The research paper outline must include an abstract, introduction, a survey of state-of-the-art tools and/or approaches on the selected topic, conclusions and future directions (if applicable). The goal is to introduce students to research methodology and teach them how to communicate technical observations and terminology professionally.

Presentations

A group of 2-3 students will be asked to give a 1-hour presentation on a selected topic from the course outline. Students must credit the original authors of the different sources of information used to prepare the presentation material by providing proper citations.

Final Exam

There will be one final exam by the end of the semester as set by the registrar's office. Unless I have reason to doubt the class is following the honor policy well, the final exam will be *open-resources exams except for human-being help*. Most of the questions on the exams will be taken directly from questions on the provided course notes, with a few additional synthesis questions.

Final Project

There will be a programming project that is due by the end of the semester. No obligation about the topic, but we will have several deadlines to progress the project in proper direction. Project will be judged by other classmates and staff based on final presentations. Collaboration of at most 3 students are allowed. Given the students show a fare amount of code sharing and participating in the project.

You are expected to commit your complete code and report to your own repository on proper revision control system (e.g., `http://github`). The committed source code project should contain the proper build scripts for compiling and building the project. Make sure to properly document your source code project and setup your revision control *repository* to be *private* from all other classmates except for explicitly announced collaborative projects.

Grading Policy

I prefer to spend my time focused as much as possible on teaching, and as little as possible on grading. The quizzes, the final exam, and the final project in this class are designed to maximize learning, rather than primarily for assessment. That said, I understand that students do need to be assigned grades at the end of the semester, and sometimes grades can be a powerful and effective motivator. Grades will be determined based on your performance on the quizzes, final project, and final exam. The grading breakup will be tentaively as follows:

Assignments	10%
Presentations	10%
Research Paper	20%
Final Project	30%
Final Exam	30%

Course Policies

Attendance Policy

Attendance in lectures is expected, and we welcome active participation (by raising questions and participating in class discussions). Please do not disturb or interrupt other classmates. You are free to leave at any time, do not feel obligated to attend the whole class.

Late Policy

Late assignments will be accepted for no penalty if a valid excuse is communicated to the instructor before the deadline. It is left to the discretion of the instructor to determine if an excuse is valid. All students are allotted a one time, no questions asked, no penalty, two day extension of exactly 48 hours from the original due date time. If a student wishes to use the one time extension the intention must be communicated to the instructor within the 48 hour window before or after the original due date time. Aside from the aforementioned exceptions, late assignments will not be graded and will result in a zero score.

Academic Integrity and Honesty

As a graduate student, you are trusted to be honorable. We will take advantage of this trust to provide a better learning environment for everyone. In particular, students in SWEN 6301 are expected to follow these rules:

- **I will not lie, cheat or steal**. If I am unsure whether something would be considered lying, cheating or stealing, I will ask before doing it.
- **I will carefully read and follow the collaboration policy on each assignment**. I will not abuse resources, including any submissions or solutions that would be clearly detrimental to my own learning.

In addition to the honor rules, students in SWEN 6301 are expected to follow these behaviors:

- **I will do what I can to help my fellow classmates learn.** Except when specifically instructed not to, this means when other students ask me for help, I will attempt to provide it. I will look at their answers and discuss what I think is good or bad about their answers. I will help others improve their work, but will not give them my answers directly. I will try to teach them what they need to know to discover solutions themselves.
- **I will ask for help.** I will make a reasonable effort to do things on my own first (or with my partners for group assignment), but will ask my classmates or the course instructor for help before getting overly frustrated.
- **I grant the course instructor permission to reproduce and distribute excerpts from my submissions for teaching purposes.** If may opt-out of this by adding a comment to your code, but without an explicit opt-out comment we assume you agree to it.
- **I will provide useful feedback.** I realize that this is a new and experimental course, and it is important that I let the course staff know what they need to improve the course. I will not wait until the end of the course to make the course staff aware of any problems. I will provide feedback either anonymously or by contacting the course staff directly. I will fill out all requested surveys honestly and thoroughly.

Additional Resources

Atlas

Atlas (https://www.ensoftcorp.com/atlas) is a program analysis framework that will be used during the course. For support with general Atlas troubleshooting issues email support@ensoftcorp.com.

GitHub

GitHub (https://github.com/) is a web-based hosting service for version control using Git.

Eclipse

Eclipse (https://www.eclipse.org/) is an integrated development environment (IDE) that will be used during the course.

CheckStyle

CheckStyle (http://checkstyle.sourceforge.net/) is a development tool to help programmers write Java code that adheres to a coding standard. It automates the process of checking Java code to spare humans of this boring (but important) task.

JaCoCo

JaCoCo (https://www.eclemma.org/jacoco/) is a Java code coverage library that will be used during the course to measure coverage for our developed test cases.

Slack Chat

A course Slack chat has been setup to allow you to seek help from your peers, teaching assistants, and the instructor. You may discuss and share some interesting topics and provide technical help to peers. The chat may also be used to provide class specific feedback to the instructor and teaching assistants.

Slack Chat: https://swen6301.slack.com/

Course Schedule

The schedule is tentative and subject to change. An up to date version of the schedule will be posted on the course website with additional materials.

Laying the Foundation ..... ≈ 3 Lectures
- Course Overview
- Clean Code
- Metaphors for Richer Understanding of Software Development
- Measure Twice, Cut Once
- Software Development Process
- Key Construction Decisions

Creating High-Quality Code ..... ≈ 2 Lectures
- Design in Construction
- Working Classes
- High Quality Routines
- Defensive Programming

Variables ..... ≈ 1 Lecture
- General Issues in Using Variables
- The Power of Variable Names
- Fundamental Data Types

Statements ..... ≈ 2 Lectures
- Organizing Straight-Line Code
- Using Conditionals
- Controlling Loops
- Table-Driven Methods

- General Control Issues

Code Improvements ..... ≈ 4 Lectures
- The Software-Quality Landscape
- Collaborative Construction
- Developer Testing and Unit Tests
- Debugging
- Refactoring
- Code-Tuning Strategies and Techniques

Concurrency ..... ≈ 1 Lecture
- The Need for Concurrency
- Concurrency Execution Models
- General Issues in Concurrency
- Testing Multi-Threaded Code

System Considerations ..... ≈ 2 Lectures
- How Program Size Affects Construction
- Managing Construction
- Integration
- The Developer Toolbox

Software Craftsmanship ..... ≈ 1 Lecture
- Layout and Style
- Self-Documenting Code