CS 341: Software Engineering I

Fall 2020

Instructor: Michael P. Rogers
Office: Halsey 214
Office Hours: MWF 2:30-3:30 PM, TuTh 11AM-12PM (virtual)
Email: rogersm@uwosh.edu
Phone: 920-424-???? / 309-825-6454

Classroom: Virtual
Class Times: TuTh 3:00-4:30 PM
Credits: 3

Prerequisites: CS 271 (data structures) with a grade of C or better, or junior-level standing and CS 262 (OOD&P II) with a grade of C or better.

Delivery: This class will be taught online

Description: This course will provide an in-depth study and analysis of at least one large scale software system. Students will analyze, design, and partially implement an extensive software project. Case studies will address major system concerns such as specification, classification, inter-relationships, validation, and evaluation. Other topics include the use of UML, prototyping, data flow diagrams, etc.

Course Website: if it happens in this course, it will be posted on UWO's Canvas site. Set up notifications to be alerted when announcements are posted, new assignments, quizzes, or notes are posted and graded, etc.

Textbook:

Required:

1. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*, 1st Edition, by Robert Martin, Prentice Hall.
2. *Debugging Teams: Better Productivity through Collaboration*, 1st Edition, by Brian W. Fitzpatrick and Ben Collins-Sussman, O'Reilly Media.

Recommended: [oreilly & kindle, respectively]

1. *Essentials of Software Engineering*, 4th Edition, by Frank Tsui, Orlando Karam and Barbara Bernal, Jones & Bartlett Learning
2. *Engineering Software as a Service: An Agile Approach Using Cloud Computing*, 1st Edition, by Armando Fox and David Patterson, Strawberry Canyon LLC

Course Outcomes :

Upon successful completion of the course, students will be able to:

1. Describe the concepts and principles that guide software engineering practice.
2. Explain the relationship between systems engineering and software engineering.
3. Define the requirements engineering process and associated activities for both computer based and web-based systems.
4. Define the analysis modeling process and associated activities for both computer-based and web-based systems.

5. Define the design engineering process and associated activities for both computer-based and web-based systems.
6. Describe testing strategies and techniques for computer-based and web-based systems.
7. Describe architectural design, styles, and patterns of computer-based and web-based systems.
8. Define the guiding principles of good user interface design.
9. Describe the use of component-level design in software engineering.
10. Describe the project management concepts applied to software engineering projects.
11. Apply software engineering principles and methods to produce a solution to a significant problem.
12. Select and apply appropriate analysis modeling practices and tools to a significant problem.
13. Select and apply appropriate requirements capture techniques and tools to a significant problem.
14. Select and apply appropriate design engineering techniques to a significant problem.
15. Design and implement an effective test strategy and plan.
16. Apply good user interface design principles to the development of a solution to a significant problem.
17. Identify appropriate situations for use of different software architectures.
18. Identify appropriate situations for use of components-based design.
19. Establish effective communication plans with clients and co-developers.
20. Conduct effective presentations for clients.
21. Conduct effective design reviews with co-developers.
22. Develop and adhere to project plans and schedules.
23. Seek timely assistance from others when individual efforts to resolve problems have been unsuccessful.
24. Establish an atmosphere of trust and mutual commitment with clients.

Grading Criteria:

Category	%
Exams (2)	30
Labs (C#, Git, etc.)	20
Reading Quizzes / Class Participation	10
Workshop Presentations	5
Team Project	35

Grade Scale:

%	≥ 92	90-92	88-90	82-88	80-82	78-80	72-78	70-72	68-70	62-68	60-62	< 60
Letter	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F

Exam Dates (tentative, subject to change):
Exam 1 - October 28
Exam 2 - December 16

Late Policies: Douglas Adams once said, "I love deadlines: I love the whooshing sound they make when they fly by". Hopefully you'll agree with the first part, and not the second, but regardless the policies are as follows:

1. Exams- If you are unable to take an exam at the prescribed time a) let me know in advance, via email, and b) provide justification (a note from medical professional who treated you, or a representative from the Dean of Students Office).
2. Labs, Reading Quizzes, Workshop Presentations - late submissions will not be accepted (plan accordingly!)
3. Project milestones - see the Team Project section, below
4.

Team Project: In this course, you will do more than just study software engineering concepts. You will put those concepts to work. In teams of 4-5, you will employ best agile practices to gather requirements, design, implement, and test a significant piece of software. The goal here is to, as best we can, simulate what happens in the workforce when teams of developers work on a product. The software is the dessert, the main course is the **process**, methodically, deliberately going through an agile development process.

You will be graded on a series of milestones and reports, and generally speaking they must be turned in by the due date. However, if you are falling behind, notify your instructor ahead of time, with an explanation of the issues and a recovery plan (fallback due date), for possible clemency. Notification after the due date will result in a 0.

Absences: It has been scientifically proven that the most significant factor for predicting student success is attendance (although whether this is truly causation or merely correlation is another question). Students should attend each and every class, and notify the instructor ahead of time if you will be absent. Attendance may be taken and factored into the Participation category.

Academic Integrity: In the same way that you can't get in shape by having your friend run round the block, or experience bliss by watching somebody else complete a crossword puzzle, you cannot learn by copy-and-pasting somebody else's answers.

First, try to solve any problem that you are presented with entirely on your own, without anything other than your text/notes. If you can do that, you truly understand the material. If not (and this definitely will happen unless you're some sort of prodigy), ask your instructor: they wrote the question, they probably know the answer :-).

You may have a *general* conversation (in English, pseudocode, or by drawing pictures) about how to approach a problem, and you may have a peer examine your code to help debug it. You may not look at or use your peer's code, nor turn in code that you find elsewhere: such plagiarism will result in a 0 for the assignment, and other unpleasantness may ensue. See the Dean of Students Office web page for more information.

Exams and quizzes are to be done entirely on your own. These are closed book, closed notes, because this material is so absolutely fundamental that you must have it at your fingertips.

Accessibility: Your instructor is committed to ensuring a fair playing field. If you have a disability and need assistance (e.g., a note taker, certain seating, extra time to take tests,

adaptive technology, etc.), please register with the Accessibility Center, and we work hard to accommodate your needs.

Non-discrimination and Anti-harassment: Your instructor is committed to maintaining a harassment-free, welcoming classroom, and will not tolerate discrimination on the basis of race, religion, creed, color, sex, gender, identity/expression, ancestry, national origin, age, marital status, relationship to other employees, sexual orientation, disability, veteran's status, membership in the military, arrest/conviction record, political affiliation, or any other protected status.

Consumer Information: https://uwosh.edu/financialaid/resources/consumer-information/