C S 40A: SOFTWARE ENGINEERING METHODOLOGIES

Foothill College Course Outline of Record

Heading	Value
Effective Term:	Summer 2021
Units:	4.5
Hours:	4 lecture, 2 laboratory per week (72 total per quarter)
Advisory:	C S 1B or 2B.
Degree & Credit Status:	Degree-Applicable Credit Course
Foothill GE:	Non-GE
Transferable:	CSU/UC
Grade Type:	Letter Grade (Request for Pass/No Pass)
Repeatability:	Not Repeatable

Student Learning Outcomes

- Use an iterative, agile process to develop a quality software product
- Design a computer program that employs the Model/View/Controller pattern

Description

A collaboration-oriented course that trains students in the techniques currently used by software engineers to develop reliable products in an efficient manner. The course emphasizes Agile methods and a variety of tools used during the software development lifecycle.

Course Objectives

The student will be able to:
A. Discuss the history of software development models.
B. Design a usable user interface for a software product.
C. Use the Agile process to develop a quality software product.
D. Understand the roles in a Scrum team.
E. Use Extreme programming techniques.
F. Design a complex software architecture that employs the Model/View/Controller pattern.
G. Use the Unified Modeling Language to design and document the relationships between different classes in an object oriented program.
H. Use Test Driven Development while coding.
I. Use a source code repository to share code with team members and integrate for testing.
J. Use an automated testing tool.
K. Use an open source code library to implement a software product.
L. Follow specific style and documentation guidelines to ensure readability and modifiability.

Course Content

A. The history and viability of software engineering methods
1. Structured programming
2. Object-oriented programming
3. Waterfall method
4. CASE tools
B. User interface design
1. User centered design
2. Prototyping
3. User testing
4. User interface guidelines
C. The Agile process
1. User stories
2. Team structure and teamwork
3. Product backlog
4. Sprints
5. Planning
6. Acceptance tests
7. Recurring delivery
8. Retrospecting
D. Roles in a Scrum team
1. Product owner
2. Development team
3. Scrum master
4. Stakeholder
5. Manager
E. Extreme Programming (XP)
1. Pairs programming
2. Continuous integration
3. Collective code ownership
4. Sustainable pace
F. Unified Modeling Language
1. Use cases
2. Collaboration diagrams
3. Class diagrams
4. State diagrams
G. Model/View/Controller
1. Design patterns
2. Application frameworks
3. Client/server
H. Software Quality Assurance
1. TDD process
2. Unit tests
3. Test frameworks
4. Refactoring
I. Version control
1. Command line and GUI
2. Local and remote repositories
3. Branch, commit, merge, clone
4. Code integration
J. Style and documentation guidelines
1. Style guidelines
2. Coding standards

Lab Content

A. Software engineering through time
1. Modify an unstructured and a structured but non-object oriented program
2. View a CASE tool
B. Design a user interface
1. Perform a user test and revise
2. Ensure adherence to user interface guidelines
C. The Agile process
1. Write user stories
2. Work in a team toward a common goal
3. Maintain a product backlog
4. Work in sprints to complete a product

5. Plan for tasks to develop in the near future
6. Write an acceptance test
7. Deliver product at multiple times throughout the development process
8. Reflect on each delivery to see what could be done better
D. Work with different roles in a Scrum team
1. Participate as a development team member on a project
2. Work with team members in other roles on a Scrum team
E. Extreme Programming
1. Practice pairs programming
2. Integrate and release code often
3. Work at a sustainable pace in a team that rises or falls together
F. Unified Modeling Language
1. Write a set of use cases for a software product
2. Read a collaboration diagram
3. Write a simple class diagram
4. Read a state diagram
G. Model/View/Controller
1. Read code that employs a design pattern
2. Use an application framework
3. Discuss tradeoffs between implementing functionality on the client as opposed to on the server
H. Software Quality Assurance
1. Test driven development
2. Unit tests
3. Test frameworks
4. Performance testing
5. Refactor code to reduce duplication
6. Automated testing tools
7. Categorize bugs
I. Version control
1. Integrate code from different engineers into one source
2. Open source code libraries
3. Use a source code control system
4. Check code into both a local and a remote repository

Special Facilities and/or Equipment

A. Access to a computer laboratory with Java and C++ compilers.
B. A website or course management system with an assignment posting component (through which all lab assignments are to be submitted) and a forum component (where students can discuss course material and receive help from the instructor). This applies to all sections, including on-campus (i.e., face-to-face) offerings.
C. When taught via Foothill Global Access on the Internet, the college will provide a fully functional and maintained course management system through which the instructor and students can interact.
D. When taught via Foothill Global Access on the Internet, students must have currently existing email accounts and ongoing access to computers with internet capabilities.

Method(s) of Evaluation

Methods of Evaluation may include but are not limited to the following:

Tests and quizzes
Written laboratory assignments which include source code, sample runs and documentation
Final examination

Method(s) of Instruction

Methods of Instruction may include but are not limited to the following:

Lectures
Online labs (for all sections, including those meeting face-to-face/on-campus), consisting of:
1. An assignment webpage located on a college-hosted course management system or other department-approved internet environment. Here, the students will review the specification of each assignment and submit their completed lab work
2. A discussion webpage located on a college-hosted course management system or other department-approved internet environment. Here, students can request assistance from the instructor and interact publicly with other class members
Team project
When course is taught fully online:
1. Instructor-authored lecture materials, handouts, syllabus, assignments, tests, and other relevant course material will be delivered through a college-hosted course management system or other department-approved internet environment
2. Additional instructional guidelines for this course are listed in the attached addendum of CS department online practices

Representative Text(s) and Other Materials

Cohn. Succeeding with Agile: Software Development Using Scrum. 2010.

Fitzpatrick, Jerry. Timeless Laws of Software Development. 2017.

Types and/or Examples of Required Reading, Writing, and Outside of Class Assignments

A. Reading
1. Textbook assigned reading averaging 30 pages per week.
2. Reading the supplied handouts and modules averaging 10 pages per week.
3. Reading online resources as directed by instructor though links pertinent to software engineering.
4. Reading library and reference material directed by instructor through course handouts.
B. Writing
1. Writing technical prose documentation that supports and describes the programs that are submitted for grades.

Discipline(s)

Computer Science