Introduction to Software Engineering

Updated version March 19, 2021

Course Information

Given at	**Chang School of Continuing Education** **Ryerson University** **Toronto, Canada**
Course code	**CCPS 406, Introduction to Software Engineering**
Term	Winter 2021
Instructor	Ilkka Kokkarinen, ikokkari@ryerson.ca
Objectives	Upon completion of the course, students will be able to: ● Develop skills that will enable them to construct high quality software that is reliable, and reasonably easy to understand and maintain. ● Understand the importance of sound engineering principles in the process of constructing software systems. ● Apply the theory of software development process in building a software project. ● Analyze system requirements using various analytical tools. ● Learn design strategies to enable efficient software construction. ● Apply testing strategies to ensure quality of software. ● Use CASE tools and software development environments. ● Practice the principles of software project management. ● Develop a team-based software system of reasonable size and complexity with a focus on managing project risks.
Evaluation	The evaluation of this course consists of four components, added together to give the total course grade: ● Project (60%, broken into smaller pieces along the way) ● Take-home final exam (40%)
Programming Project	Students will design, implement, test and release a reasonably sized programming project, from the vision document provided by the instructor who then acts as the customer of this project. Each project must be done in groups of three to four students formed during the first week All project code and documents must be maintained under Git version control and stored in GitHub. The project grade will be calculated from the list of required milestone

	reports and the final presentation. In the Winter 2021 term, this project will be a small interactive fiction (aka "text adventure") game, consisting of a simple game engine and enough gameworld data to demonstrate the behaviour of the engine. The setting, theme and storyline of this game are to be freely chosen by the group.

Material

SE10	Ian Sommerville: Software Engineering, 10th Edition. The official textbook of this course, along with its collections of lecture slides, lecture videos and supplemental material. We will only cover the first nine chapters of this book during this first course, leaving the topics of the remaining chapters for more specialized courses on software engineering. This textbook can be ordered from Amazon and other places where fine books are sold. The publisher-approved eText version is also available for either one semester or lifetime rental.
SM	Sourcemaking. An excellent collection of tutorials on software design patterns, antipatterns, UML diagrams and software refactoring.
TCC	The Codeless Code. Darkly humorous fables on software engineering for your right brain, illustrating principles of programming philosophy and its best practices. Written as an affectionate parody of zen koans, the reader is expected to connect the implied dots to lift these fables into the realm of modern software engineering.
MB	Martin Fowler. Weblog of Martin Fowler's observations on agile software design and related topics.
CCB	Clean Coder Blog. Weblog of Robert Martin's observations on object-oriented design, programming and testing.
EIP	Epigrams in Programming. Despite being four decades old, these fundamental observations on computer programming remain as timeless as ever.
YouTube	A collection of videos, selected by the instructor, where various colourful characters of this field provide their views, opinions and experiences with various topics of this course.

Course Modules

Module 1: Introduction

YouTube	Ian Sommerville: "10 Questions to Introduce Software Engineering" John Osterhaut: "A Philosophy of Software Engineering" Traversy Media: "Git & GitHub Crash Course for Beginners"
SE10	1: Introduction
TCC	Case 162. All according to plan Case 180. Past perfect Case 187. Exit interview Case 207. Parrot Case 229. The recommended approach
MF	Technical Debt Technical Debt Quadrant
CCB	What software craftsmanship is about The obligation of the programmer Where is the foreman?

Module 2: Software Processes

YouTube	Ian Sommerville: "Plan-based and Agile software processes" Genesis Consulting: "Agile vs. Waterfall" Randy Rice: "V-Model" Martin Fowler: "Continuous Delivery"
SE10	2: Software Processes 3: Agile Development
TCC	Case 10. Pride Case 22. Safety Case 100. Ten thousand mistakes Case 159. Blocks Case 188. Tough love
MF	Waterfall Process Principles of XP Objects and Iteration
CCB	Why is estimating so hard? The churn The start-up trap

Module 3: System Modeling

YouTube	Derek Banas: "UML 2.0 Introduction" Derek Banas: "UML 2.0 Activity diagrams" Karl Wiegers: "What is a use case?"
SE10	5: System modeling
TCC	Case 70. System Case 137. A thousand words Case 154. A bridge to nowhere Case 168. Horizontal, vertical Case 189. The dense forest
MF	Domain Driven Design Software Development Attitude
CCB	Solid relevance Agile is not now, nor was it ever, Waterfall Why won't it...

Module 4: Software Design

YouTube	LucidCharts: "UML class diagram tutorial" LucidCharts: "UML use case diagram tutorial" Karl Wiegers: "Use cases and functional requirements" Derek Banas: "UML 2.0 Sequence Diagrams"
SE10	6: Architectural design
TCC	Case 1. The small stuff Case 64. Three beggars Case 86. Consequences Case 212. Seasons Case 230. All together now
MF	When To Make a Type? Function Length Beck's Design Rules
CCB	Screaming architecture Make the magic go away The frenzied panic of rushing

Module 5: Design and Implementation

YouTube	Sandi Metz: "SOLID Object Oriented Design" Saasbook: "Patterns, Antipatterns and SOLID Class Architecture"

	Christopher Okhravi's Code Walks: "Single Responsibility Principle", "Liskov Substitution Principle", "Interface Segregation Principle", "Dependency Inversion"
SE10	7: Design and implementation
TCC	Case 46. Jinyu's Tack Case 104. Guardrails Case 111. Labyrinth Case 175. Flat shark Case 214. Streams of consciousness
MF	Yagni Continuous Integration On Pair Programming
CCB	Future proof Code hoarders Classes vs. data structures Tools are not the answer

Module 6: Design Patterns

YouTube	Jack Herrington: "Five Design Patterns Every Engineer Should Know" Derek Banas: "Design Patterns Video Tutorial" (you don't need to watch this entire playlist, just the introduction followed by the individual videos about the particular design patterns that you want to learn more about)
SM	Creational: Builder, Factory Method, Object Pool Structural: Decorator, Adapter, Facade Behavioural: Iterator, Observer
TCC	Case 15. Immutable Case 98. Anti matter Case 155. Don't help Case 156. The garden path Case 195. The magician's code
MF	Is Design Dead? Getter Eradicator Uniform Access Principle
CCB	Pattern pushers The single responsibility principle The open-closed principle

Module 7: Software Validation

YouTube	Philip Johnson: "A beginner's guide to testing" Development That Pays: "What is Test Driven Development?" Development That Pays: "TDD and BDD: What are the key differences?"
SE10	8: Software testing
TCC	Case 103. The black enamel box Case 135. Ass-backwards compatibility Case 142. The blind leading the blind Case 182. Mousetrap Case 219. Nothing really matters
MF	Test Categories Unit Test Test Cancer Mocks Aren't Stubs Object Mother
CCB	TDD harms architecture When TDD doesn't work Giving up on TDD

Module 8: Software Antipatterns

YouTube	Andy Sterkowitz: "Five Programming Antipatterns for Beginners" Martin Fowler: "Software Design in the 21st Century"
SM	Code smells Refactoring techniques
TCC	Case 41. Garbage Case 84. What it says on the tin Case 115. Pain Case 121. Where angels fear to thread Case 209. Submerged
MF	AntiPattern Workflows of Refactoring
CCB	Thorns around the gold The dark path Types and tests

Module 9: Refactoring

YouTube	Coding Tech: Learn Code Smells and Level Up Your Game!

	Frederick Vandrabant: The Law of Demeter: Tell, Don't Ask Christopher Okrhavi's Code Walks: Law of Demeter
TCC	Case 94. Conventions Case 118. Clay Case 178. Unity Case 210. Hygiene Case 213. The imperfect mirror
MF	Refactoring
Others	Things of Interest: It is probably time to stop recommending Clean Code DZone: The Genius of the Law of Demeter

Module 10: Software Evolution

YouTube	Christina Hill: "The 4 Types of Software Maintenance" Loop Conf: "Zen and the Art of Software Maintenance" Christopher Okhravi: "E-Type, P-Type, S-Type Systems"
SE10	9: Software Evolution
TCC	Case 35. Sparrow breaks free Case 78. Crashing the third gate Case 128. The prison of infinite pleasures Case 133. Dead language Case 134. Thin ice Case 228. The trembling giant
MF	Canary Release Frequency Reduces Difficulty
CCB	A little structure Stabilization phases Is Doctor Calvin in the room?

Module 11: Project Presentations

During this week, student groups publicly present their projects to the instructor/customer in front of their peers. Discussion of these projects and their successfully implemented features as seen from the user's point of view.

Module 12: Project Postmortems

During this week, student groups report and present their learning experiences on software engineering. Discussion of ways that the engineering issues and topics discussed during this course came up during the group project, and what human, communications or other outside issues affected the productivity of the group during the project.

Module 13: Final Exam

The format of the final exam is a take-home exam that consists of essay questions, to be completed individually and independently by each student. You have the entire final exam week to think up and write down your answers before submitting them as a single document.