CSCI 4210: Introduction to Software Engineering

University of New Orleans
Department of Computer Science

Meeting Time and Location

- **Times:** 3:30 pm - 4:45 pm
- **Days:** Tuesday/Thursday
- **Room:** Math 322 (Lecture)
- **Discord:** https://discord.gg/mWUEMB2jET

Instructor Information

- **Instructor**: Ted Holmberg, Ph.D.
- **Office**: Math 347
- **Email**: eholmber@uno.edu
- **Office Hours**: By Appointment and Monday/Thursdays from 5:00 pm to 6:30 pm. or Online chat via Discord.

Student Information

Students should meet the following prerequisites and expectations:

- Completion of CSCI 2125 (Data Structures) or an equivalent course.
- Strong programming and problem-solving skills.
- A strong work ethic and openness to learning through documentation and research.
- Willingness to participate in teamwork and peer-based collaboration.

If you're not comfortable sharing your ideas, asking questions, working in groups, speaking up in class, or figuring out things on your own, then this is NOT the course for you.

Course Overview

This course provides an introduction to software engineering, focusing on life cycle methods and tools. It emphasizes the design, implementation, and testing of Object-Oriented (OO) systems, as well as general software development practices. A central component of the course is a team-based project that enables students to apply their knowledge in a practical context.

Topics Covered

The following is a tentative list of topics that will be covered throughout the course:

- Introduction to software engineering and the software development life cycle
- Analysis of specifications and design in OO systems
- Key principles of software design:
  - Cohesion and coupling of classes
  - Single Responsibility Principle
  - Open-Close Principle
  - Dependency Inversion Principle
  - Liskov Substitutability Principle
  - Interface Segregation Principle
  - Narrow Interface Principle
- Refactoring and design patterns:
  - Simple Interface, Abstraction, Interface, Composition, Design by Primitives, Narrow Interface, Factory, Flyweight, and other advanced patterns
- Testing methodologies:
  - Unit Testing, Automated Testing, Mock Objects
- Software configuration management and version control
- Frameworks as reusable software products
- Introduction to software architecture:
  - MVC-driven architecture, Pipes and Filters
  - Fundamental architectural patterns

The course content is flexible and may be adjusted based on student interests. Topics may be added or removed as needed.

Course Objectives/Outcomes

This course explores the core concepts of software engineering, emphasizing teamwork, collaboration, and communication. Through practical application, students will develop both technical skills and soft skills, both of which are critical to success as a software engineer.

Course Objectives

The course objectives focus on the concepts and methodologies that students will learn and understand:

- Understand the software development process, including management, risks, and key factors involved.
- Learn to develop and manage use cases and software life-cycle models for project development.
- Analyze and resolve design issues in software development, with a focus on the role of frameworks in software design.
- Improve communication and presentation skills, emphasizing the clear articulation of technical concepts.
- Grasp version control practices, including committing, branching, merging, and resolving conflicts.
- Explore and practice agile and iterative development methodologies, focusing on teamwork, collaboration, and constructive feedback.
- Learn to read and interpret technical documentation to understand and use new tools and technologies.
- Develop self-learning strategies to adapt to new tools and technologies as needed for project work or industry demands.

Expected Outcomes

The expected outcomes focus on the tangible deliverables and skills that students will produce by the end of the course:

- Deliver a comprehensive set of requirements and use cases to design software systems.
- Implement, test, and refactor software systems following best practices.
- Complete a full software development lifecycle, including problem analysis, design, implementation, testing, and refactoring within a team environment.
- Present and communicate technical solutions effectively in both written and oral formats.
- Manage and integrate code within a version control system, ensuring seamless collaboration and version management.
- Produce a collaborative project using agile methodologies, demonstrating the ability to work within a team and integrate peer feedback.
- Apply technical documentation to successfully learn and utilize new tools and technologies in software projects.

Pedagogical Approach

Industry-Like Experience: This course simulates a real-world professional environment, providing students with an industry-like experience where collaboration and communication are key. The entire class will function as a startup organization, with all students working together under the banner of 4210-Startup. The class will take on clients (stakeholders) of my choosing who will provide various work requests with specific project requirements. The job of the class is to deliver on all these requests, mirroring the dynamics of a startup taking on multiple projects.

Pod Structure and Team Dynamics: Students will be grouped into teams of three, referred to as pods. Each pod will be responsible for a specific aspect of the overall client work. These pods will need to work both internally and externally:

- **Intra-Teamwork (Within Pods)**: Within each pod, students will collaborate closely to meet their internal goals and deliverables. This will require effective teamwork, communication, and division of responsibilities among pod members.

- **Inter-Teamwork (Between Pods)**: Pods will also need to interact with other pods as part of the broader startup organization. Some pods may serve as clients to others, requiring clear communication and coordination to ensure that the various components of the client requests are integrated into a cohesive final product. This inter-teamwork is essential for ensuring that all client needs are met effectively.

- **Client-Provider Relationships**: Certain pods may take on the role of clients, requesting specific deliverables or services from other pods. This mimics real-world scenarios where different teams within an organization must interface and collaborate to achieve a shared goal, particularly in responding to client needs. To enable effective collaboration and ensure that multiple pods can work concurrently, even in the absence of completed work from other teams, the course will motivate the adoption of strategies such as Test-Driven Development (TDD). TDD allows pods to define clear interfaces and write tests in advance, enabling parallel development and reducing dependencies on the completion of other pods' tasks.

Individual Evaluation: While the work is team-based, individual student performance will be assessed based on their sprint assignments and commit history. Each student's contributions will be tracked through the tasks they complete during sprints and the code they commit to the repository. This ensures that individual effort is recognized and that each student's grade reflects their personal involvement in the project.

Client-Focused Delivery: The ultimate goal of this course is for the entire class to come together as a cohesive unit to build, integrate, and deliver work that meets all the requests taken on by 4210 Startup. The goal is for students to cultivate the ability to function within and across teams, preparing them for the software engineering industry.

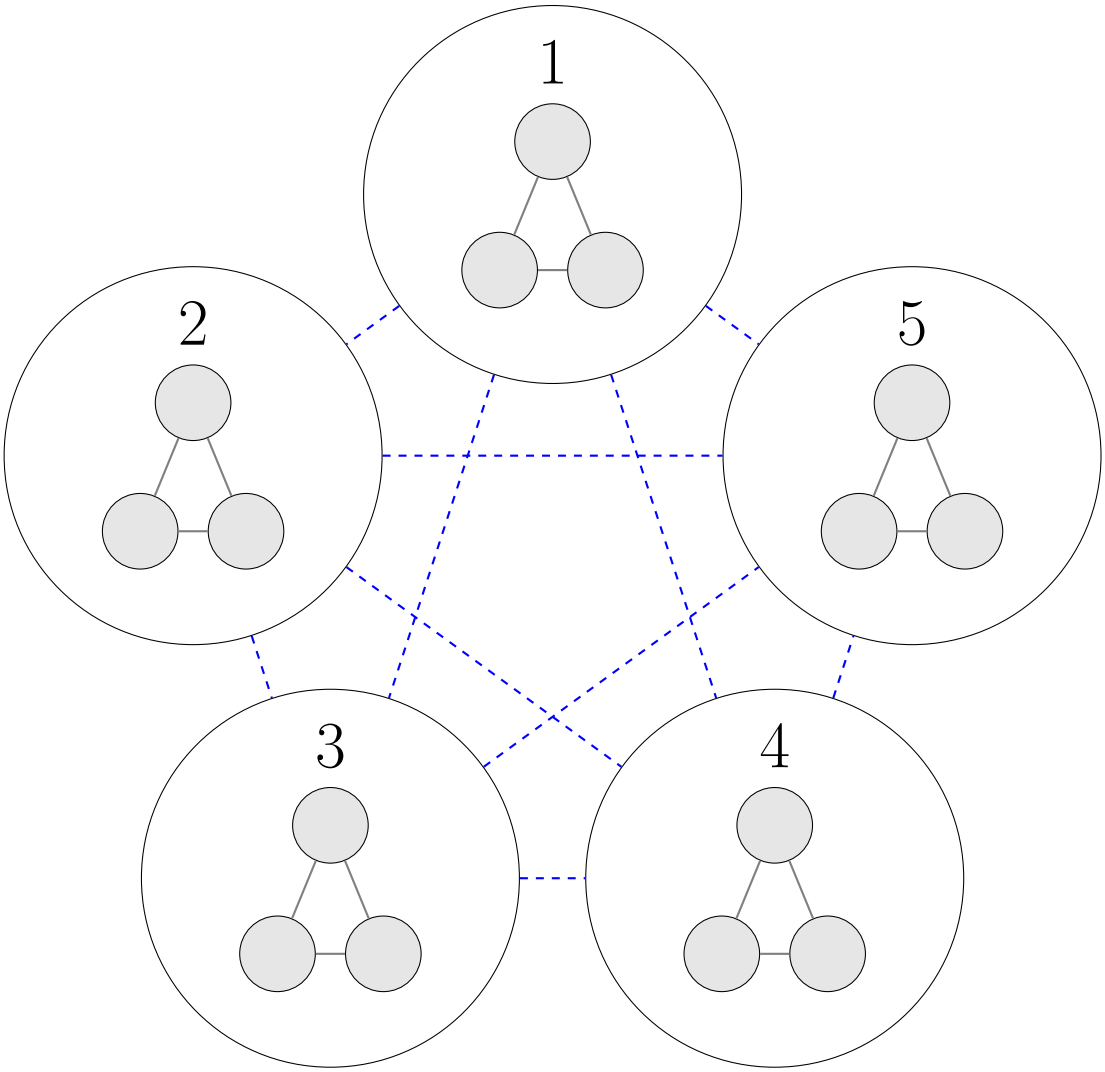


Figure 1: Intra-teamwork within each pod (gray lines) and inter-teamwork across pods (blue dashed lines), with each pod consisting of three individuals.

Graded Components

Students are expected to actively participate in various components throughout the course, each designed to build and reinforce their software engineering skills. The major components include workshop development, lab completion, code refactoring, and a final team project.

---

1. Tech Workshop Development

Subject Matter Experts (SMEs) Each team is responsible for developing and presenting a tech workshop to the class, which includes walking through the setup and usage of their assigned technology.

Ongoing Technical Support This workshop must include a live walkthrough for their peers to follow during the presentation. After the workshop, that team will then be responsible for supporting that technology for their peers for the remainder of the semester.

Deliverables:

- A developed lab workshop on the assigned technology.
- Comprehensive documentation and a walkthrough for peers.
- Ongoing support for the technology throughout the course.

Note: The first team to present will receive bonus points for taking the initiative.

---

2. Tool Certification

Students are required to attend and successfully complete all workshops created by their peers. These workshops will be treated as a certification process, with each student responsible for mastering and demonstrating proficiency in the tools and technologies presented by other teams.

Deliverables:

- Certification of completion for all assigned workshops.
- Submission of any required outputs or results as specified in the workshop instructions.
- Demonstrated proficiency in the use of these technologies in the development and delivery of the final project.

---

3. Code Refactoring Assignment

Each student will individually refactor a previous project, bringing it up to professional standards. This assignment includes a presentation and a peer review session.

Objective: To develop students' ability to improve existing code and align it with professional standards.

Activities:

- Refactor a previous homework assignment into a polished application.

- Improve code quality, add proper documentation, create a professional GitHub repository, and consider integrating a front-end or API documentation.
- Participate in weekly stand-ups to present progress and receive peer feedback.

Deliverables:

- Refactored codebase submitted for grading.
- A professional GitHub repository with documentation.
- Presentation explaining the refactoring process and improvements.
- Peer evaluations reflecting the quality of code and documentation improvements.

---

4. Final Project

The final project grade will be based on each student's individual contributions to the team's progress, as tracked through completed sprints. Students will work in teams on software development projects and apply advanced software engineering methodologies. Each student's grade will reflect their personal effort and achievements within the sprints, ensuring accountability and individual assessment.

Sprint Tracking: Students will participate in regular stand-ups to present their individual progress based on assigned sprint tasks throughout the semester. Each student's work will be tracked using a project management tool, such as GitHub Projects or a similar platform, allowing for detailed assessment of their contributions.

Deliverables:

- Regular sprint updates and completion of assigned tasks.
- Contributions tracked via commit history and project documentation.
- Final project submission, integrating all pods' contributions.

Grading

This course is designed to be project-oriented, with a strong emphasis on hands-on learning and practical application. Your grade will reflect your engagement in lab creation, completion, individual assignments, and the final team project. The grading breakdown is as follows:

Graded Component	**Percentage**
Tech Workshop Development	20%
Tool Certification	20%
Code Refactoring Assignment	20%
Final Project	40%

Letter Grades

The grading scale is as follows:

Letter Grade	**Percentage Range**
A	≥ 90
B	80 - 89.99
C	70 - 79.99
D	60 - 69.99
F	< 60

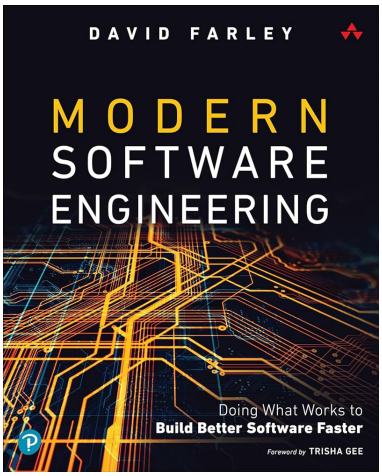
Student Success

Grading is done as objectively as possible using clear metrics and rubrics discussed in class. If you put in the effort and participate, you will earn a good grade in this course.

Textbooks and Other Materials

Required

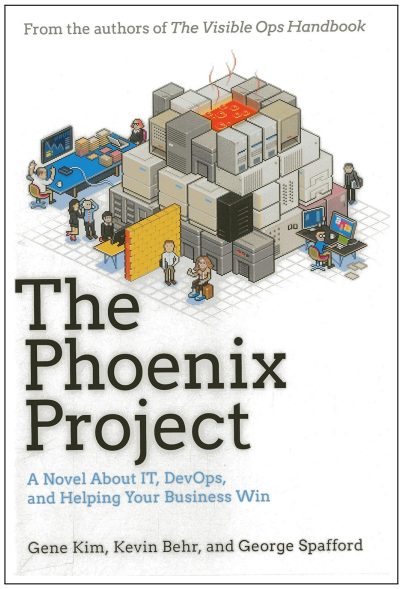
Title: *Modern Software Engineering: Doing What Works to Build Better Software Faster*
Author: David Farley
ISBN-13: 978-0137314911

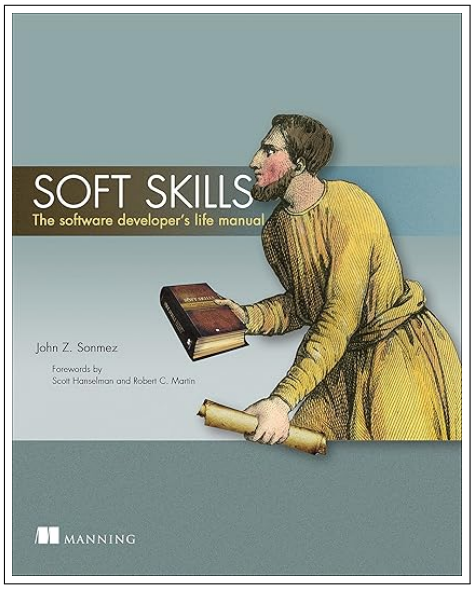


Recommended

1. **Title:** *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win*
   Authors: Gene Kim, Kevin Behr, George Spafford
   ISBN-13: 978-0988262591

2. **Title:** *Soft Skills: The Software Developer's Life Manual*
   Author: John Sonmez
   ISBN-13: 978-1617292392





Course Communications

Throughout the course of this semester, we will communicate with one another in person as well as via Discord. In every interaction throughout the semester, regardless of the medium, it is my expectation that we all communicate professionally and respectfully.

Course Policies

Attendance

Students are expected to attend classes on time and participate in class activities. Attendance will be taken at each class meeting. Important course content is often introduced outside of the published/provided sources and/or scheduled presentations.

Academic Integrity

Work hard and learn useful skills and knowledges. Otherwise, you are wasting your time.

Students with Disabilities

It is my intent to fairly serve all students in this course. I will make every effort to facilitate reasonable accommodations for any student with a documented illness or disability. With respect to academic fairness, non-trivial accommodations must be approved by the University's Office of Disability Services.