CS 422/522 – CRN 11608,11619
Software Methodologies I
Fall 2025

1. General Course Information

UO Catalog Description

Prerequisites	CS 313 - Intermediate Data Structures.
Lectures	M W, 12:00 – 01:30, B040 PSC (Basement, Allan Price Science Commons & Research Library – see map https://library.uoregon.edu/scilib/map)
Description	This course includes technical and non-technical aspects of software development, including specification, planning, design, development, management, and maintenance. Students must complete two group projects.

Teaching Team

Juan Flores (Instructor)
he/him/his
Office: 158 Deschutes Hall
Email: jflore10@uoregon.edu
Zoom: https://uoregon.zoom.us/my/juan.cs.uo

Hritvik Jekki (GE)
he/him/his
Office: 228 Deschutes Hall
E-mail: hritvik@uoregon.edu
Zoom: https://uoregon.zoom.us/my/hritvik

This term, our office hours will be on demand; email us if you need to see us.

Course Overview

This course teaches you methodologies and practices for building adaptable, robust, reliable, and usable software systems. Developing such systems requires the cooperation and participation of several team members. This class will help you learn how to work effectively as part of a team. You will learn structured approaches for analyzing systems requirements, specifying software design, testing systems, and managing the software development process. This term, we will study how to apply AI across the phases of software and Systems development.

Textbook

1. Sethi, Ravi. *Software Engineering: Basic Principles and Best Practices*. Cambridge University Press, 2023. ISBN: 9781316511947 – ***Required***. A free version of the book is available at the Science Library (https://alliance-uoregon.primo.exlibrisgroup.com/discovery/fulldisplay?context=L&vid=01ALLIANCE_UO:UO&search_scope=Everything&tab=Rollup&docid=alma99901218360501852)

Learning Outcomes

After completing CS 422, you will be able to:

- Design and construct quality software systems within schedule and resource constraints.
- Make design decisions that effectively balance multiple goals and constraints. Constraints include schedule, resources, usability, accessibility, maintainability, testability, efficiency, as well as internal and external risks.
- Understand the relationship between product structure (software architecture) and team structure.
    - Assess the consequences of a dependency in code for communication among team members.
    - Modularize software design to accommodate division of work among team members
- Understand the relationship between product structure and team development processes.
    - Create a development plan that supports parallel work and controls risk by providing regular progress checks.
    - Assess and prioritize risks. Select risk mitigation tactics tailored to a project, including those influenced by factors beyond the development team's control.
- Work effectively as a team.
    - Foster confidence and commitment among team members.
    - Hold effective, time-efficient meetings.
    - Maintain an appropriate level of awareness of teammates' activities.
- Use AI tools to assist you in the software development process, from design to coding.

While our primary learning objectives lay a foundation for lifelong learning, you will also acquire essential skills in contemporary approaches and technologies. These include:

- Programming. You may choose a programming language based on its suitability for solving each project.
- Development environment. You may use any IDE you prefer or any other set of tools if appropriate.
- Debugging. You must learn to debug effectively using the appropriate tools.
- Version control. Every team must use git.
- Testing. You will automate testing to a practical extent; use unit testing tools appropriate for your programming language.
- Incremental development. While we will not adopt any particular methodology, our approach will loosely follow agile methods in general and *Scrum* in particular, with influence from the *spiral model* for risk-based planning and monitoring.
- Documentation. Your product must include concise yet practical documentation for both developers and end users.
- Mastery of AI tools, e.g., GitHub Copilot, LLMs (ChatGPT, Claude, Gemini, and others), Cursor, MCP (Model Context Protocol), Spec Kit, etc.

Credit Hours and Student Workload

The following is an example of the expected workload for an undergraduate student. Graduate students must do roughly a third more work than their undergraduate counterparts.

Educational Format or Activity	Hours	Explanation/Justification
Lecture (80-minute session)	30	20 lectures @ 1.5 hrs
Discussion sections	10	1 hr./week - project progress meetings and office hours
Assigned readings	30	~25 pages (approx. 3 hrs) per week
Project work	50	5 hrs./ week - analysis, design, programming, etc.
TOTAL HOURS	120	

Read, study, and work on your project regularly throughout the course. That way, you will not experience massive overloads before exams or when projects come due.

2. Communication

Communicate Questions and Concerns

Please contact the instructor with any questions or concerns about the class. In your communication, please follow these guidelines:

- Follow the UO Information Services netiquette guidelines (https://classes.cs.uoregon.edu/20S/cis422/Netiquette.html).
- Check your @uoregon.edu and @cs.uoregon.edu email accounts daily, consistent with the Registrar's email policy (https://registrar.uoregon.edu/current-students/student-e-mail-communication-policy).
- Send all course-related emails from your UO email accounts, consistent with the Registrar's email policy.
- If necessary, you can forward your cs and uoregon email (https://systems.cs.uoregon.edu/wiki/index.php?n=Help.Email and https://service.uoregon.edu/TDClient/2030/Portal/KB/ArticleDet?ID=30799).
- Configure your email client to display your first and last name in the "From:" header.
- Include "422" in your email's subject line and describe the topic briefly.
- Keep signature files to a minimum.
- Do not send advertisements.

Students are responsible for communicating with the instructor about any questions, concerns, or issues that arise during the course. These may involve any aspect of the class, including lectures, group dynamics, and communication breakdowns. Please visit the instructor during office hours or schedule an appointment to discuss any problems or concerns.

Canvas and Email

Your instructor will post announcements on the Canvas platform. This mechanism intends to reach the whole group.

You must communicate with your teaching team through direct email. Please do not use Canvas to contact your instructor/GE. *I will not reply to any Canvas messages*.

Use the most appropriate communication channel according to the situation.

(1) Use Email for questions about grading, advising questions, etc.
(2) For questions that require more time, office hours are the best option. Send an email describing your issue(s) in advance.
(3) Students must address all grading discrepancies via email. No Canvas, please.
(4) You are encouraged to contact me in person (after class, during office hours, or by appointment) or by email.

3. Course Resources

Attendance and Participation

Lecture attendance is not mandatory, but your participation in presentations and discussions will enrich the learning experience for everyone. Software design and development are iterative processes that often involve group discussions of design alternatives. The instructor or students may present submitted work to the class for critique and debate. Your in-class participation may be supplemented by emailing the instructor with questions or comments.

Online Resources

Please log in to Canvas (https://canvas.uoregon.edu/) to access all course materials. I will publish slides and accompanying materials (organized into modules) before each week begins. It is advisable to keep a personal copy of the lecture notes so you can add your own annotations.

You may find the following websites helpful:

- Remote Resources for UO Students: https://remote.uoregon.edu/student
- Accessible Education Center: https://aec.uoregon.edu/

Lecture notes are available for download from the course Canvas website. You can also annotate printouts with your own handwritten or digital notes.

Personal Computer

Regardless of the computing resources the University provides, you must have a personal computer to work on the projects. This computer should have at least 3 GB of free hard drive space for this class and an up-to-date operating system. You should possess the minimum system administration skills to manage your system and maintain secure, reliable backups of your coursework. If your computer crashes during the term, you can replace it and quickly reinstall the required software, programs, and data.

Computer Usage while in Class

During class, you are welcome and encouraged to use your computer to take notes, edit and test code, and perform other class-related activities. Please refrain from using social networks, watching videos, playing games, engaging in sports, and other distractions while class is in session.

4. Evaluation

Group Project

In this course, you are required to participate in and complete one eight-week group project. Each group must propose a project topic and select its team members. The project documentation (on Canvas) outlines the minimum project requirements.

You must be proactive and participate in group meetings outside of class. Be generous and flexible with your time when scheduling team meetings. Maintain constant communication with group members.

A designated team member will submit the required products for each project stage (i.e., partial deliveries) via email to the instructor and GE. Ensure all documents include the names of all team members.

Evaluation Criteria

We will grade according to the evaluation criteria I will distribute with the project. Read these criteria carefully, as they reflect the crucial aspects; if the solution your team submits meets those criteria, you will succeed in this course. We will use a rubric worksheet to grade your project; locate it on Canvas > Files > Project.

Subjective Assessment

Computer science is generally objective, for example, in determining whether a computer program will compile and produce a specified output or fail to do so. Nonetheless, most of this class's material comprises concepts, ideas, terminology, conventions, and practices that cannot be defined in purely objective language, such as that used in computer programs. Exams, quizzes, and the project are graded based on the instructor's subjective assessment of the accuracy and completeness of students' answers and the quality of the software they produce.

Late Projects

Late submissions will incur a 10% grade penalty. Late projects will not be accepted after 48 hours past the due date and time. If needed, students must request a justified exception before the due date.

Grading Discrepancies

If you have any grading discrepancies (such as miscounted points on an assignment), please request clarification or a correction via email within a week of the project or exam grade being published.

Exams

The midterm and the final will be the same length and format, consisting of 30 1-point questions. Both exams are in person, closed-book, with no internet searches or electronic communication.

Missed Exams

You should provide a valid reason for missing an exam. Contact your instructor in *advance* to schedule a time to take the missed exam.

Video

Each team must produce a short video describing their project and the tools, resources, and techniques used. The video must highlight the product's characteristics and demonstrate how the project advances the state of the art in the application area, software engineering, or any other relevant field where it may make significant contributions. It is also essential to mention what you learned while developing the project.

I encourage you to pay close attention to the video's production and include it in your work portfolio. It will be an asset in your job and/or internship search.

Computing your Course Grade

Your final grade will be weighted as follows:

Component	Percentage
Exam 1	20
Exam 2	20
Project	50
Video	5
Participation	5

The final letter grade will be computed using a straight percentage (not a curve), according to the following table:

Letter grade	Percentage	GPA
A+	97–100%	4.33 or 4.00
A	93–96%	4.00
A–	90–92%	3.67
B+	87–89%	3.33
B	83–86%	3.00
B–	80–82%	2.67
C+	77–79%	2.33
C	73–76%	2.00
C-	70–72%	1.67
D+	67–69%	1.33
D	63–66%	1.00
D-	60–62%	0.67
F	0–59%	0.00

The quality of your participation in progress report meetings, class engagement, and class presentations (optional) determines your class participation grade component.

5. Product Quality

Produce Good Writing

The instructor will partially evaluate projects based on his subjective assessment of the quality of the submitted written materials. A technology expert must be able to communicate ideas clearly and concisely.
Good writing occurs on three levels:

1. Structure a piece of writing so that the main ideas are accessible. State the main point of a paper in the introduction. Begin each paragraph with a topic sentence. Break the report or document into sections and assign a title to each one. Summarize your key findings in the conclusion. Stream-of-consciousness storytelling does not lend itself to good organization.
2. Communicate ideas effectively. Be thorough but concise. Your tone must be formal and direct, as if you were reporting to your boss in a real job. An informal "chatty" style is not appropriate. Every figure (graph, drawing, or screenshot) must be relevant, referenced in the main body of the text, have a caption that starts with "Figure *x*.", and explain what the figure shows.
3. All spelling and grammar must be standard and correct.

Organize your ideas in paragraphs, but do not write long run-on paragraphs. Always proofread your work. Do not write paragraphs that are lists with lots of text jammed together.

Actively work to improve your writing. Take writing courses and study books such as The *New Oxford Guide to Writing* and Strunk & White's *Elements of Style*. Reach out for help at the Tutoring and Engagement Center (https://engage.uoregon.edu/).

Produce Good Media

Any media you submit for coursework should be of high quality. Photographs should be sharp, properly exposed, and cropped, and they should effectively communicate what you intend to express. Videos should adhere to the same quality standards. Data files should be as large as needed to deliver the relevant content.

Every team must produce a brief video highlighting the system they have developed. The video must be between 5 and 15 minutes long.

6. Course Policies

Academic Policies

Universal Learning Environment – The University of Oregon, the CS department, and your instructional team support inclusive learning environments. Please notify your instructor if any aspects of the course's instruction or design hinder your participation.

General UO Policies

Canvas offers a menu item that opens a web page listing policies, explaining them, and providing additional information and links (e.g., academic accommodations, campus disruptions, and personal support).

Academic Integrity.

This section complements the University Course Policies. This complement is critical, given that we will emphasize and encourage the use of AI in Software Engineering.

All UO students should be familiar with the UO Student Conduct Code. Read this document https://policies.uoregon.edu/vol-3-administration-student-affairs/ch-1-conduct/student-conduct-code, besides the Academic Honesty section of the University Course policies found on Canvas.

Academic *honesty* includes the following. You *should* do all of the following:

- List each person and their contribution to any project or exam. Include this information in all submitted documents and in the header of all affected source code files. The contributor might be a tutor, roommate, officemate, fellow student, or any other person who contributed to your work.
- Identify and delimit any content (such as written text, computer code, photographs, videos, and sounds) derived from another source, whether published or unpublished, and indicate the source and author in the computer code, user interface, and documents.
- Document all use of AI, including the AI assistant you used, the prompts or spec files you provided, and so on.
- Be prepared to explain any program code you submit.

Academic *dishonesty* includes the following. You should *not* do any of the following:

- Misrepresent someone else's work, including anything you find on the web, as your contribution or result, or in any way contribute to such a misrepresentation.
- Submit someone else's work as your own without citing the source, including code generated with AI assistance.
- Knowingly or accidentally, make your work available to another student so they can submit it as their own work.
- Forge a document that you then use in an academic setting.

I will pursue all evidence of academic dishonesty in accordance with the guidelines in the Faculty Guide for Addressing Suspected Academic Misconduct (https://dos.uoregon.edu/files/faculty-guide.pdf).

There is a crucial point or exception regarding academic honesty: You are allowed and encouraged to use AI agents to assist with project development. Please indicate in your project documentation how much help you received from these resources. Please specify the agent or system you used, including the prompts and their responses, as well as your interaction with them.

Tips for Academic Success.

i. Know the course's Academic Policies.
ii. Monitor your teammates' involvement, participation, and progress. If you notice that any member is not working, discuss the issue with the team and notify the instructor immediately to address the situation.
iii. Submit your project on time; you will not pass the course without completing it.
iv. Monitor your grades in the Canvas gradebook. Notify your instructor of any problems as soon as possible.
v. If you are having difficulty in the course, please see your instructor sooner rather than later, while we still have recovery options available.