CSDS 393/493: Software Engineering

Spring 2025

Tuesday and Thursday: 2:30 PM – 3:45 PM | Nord Hall 410

Course overview: Modern software engineering encompasses both well-founded software development practices and the area of computer science and engineering that deals with the specification, design, implementation, validation, and maintenance of complex software systems and applications. Successful software projects require more than just technical expertise. Figuring out what the client wants, collaborating in a team, managing complexity, mitigating risks, staying on time and budget, and determining when a product is good enough to be shipped are at least equally important topics. This course covers the principle elements of software engineering methodology and has a strong technical focus including assignments with and without programming. Students will get experience with team management and modern software-engineering tools through working on a significant team project.

Objectives:

- Understanding of the software development lifecycle
- Understanding of the issues and challenges associated with each phase of software development
- Familiarity with the basic software engineering techniques and tools for each phase of software development
- Experience working as part of a team on a significant software development project
- Cultivation of critical thinking skills needed by professional software engineers

Pre-requisites:

- CSDS 233 or ECSE 233 or Data Structures with a grade of C or higher
- Intermediate to advanced programming skills in at least one modern programming language, such as Java, C++, C#, Python, Rust, Kotlin
- Computer Science Major/Minor or Data Science Major (for CSDS 393 and CSDS 493 students)

Instructor: Sumon Biswas
Email: sumon@case.edu
Office: 608 Olin Building
Office hours: TBD (see canvas)
TAs: TBD (see canvas)
Canvas: https://canvas.case.edu/courses/47073

Textbook: There is no required textbook, rather we have assembled readings from different sources. Online readings will be posted on Canvas. If you want to get one to supplement the class notes, I recommend the latest edition of *Software Engineering* by Ian Sommerville (Pearson publishers) though it does not cover exactly the same material. As an optional supplementary reading, consider the (freely available) Software Engineering at Google, Lessons Learned from Programming Over Time.

Canvas: Presentation slides, announcements, and other materials will be posted on Canvas. However, these are not a substitute for attending class. You are responsible for what is discussed in class, whether or not it appears in the slides. Almost every topic discussed in class is also discussed on various software engineering websites and on YouTube, which you are encouraged to consult for more information and different perspectives. In case of conflicting interpretations of a concept, the one given in class should be considered authoritative for quizzes and assignments.

Grading:
- Quizzes and homeworks (60%)
- Project (40%)

Late work policy: No makeup quiz or late assignment submission will be allowed unless either: (1) you request at least 48 hours before the quiz or submission date to be excused from class (giving a valid reason) and I approve your request or (2) you are sick on the day a quiz or assignment is due and you provide me with a note from a health care provider confirming that. To avoid a grade of 0 you must take a makeup quiz or turn in the relevant assignment within two days of returning to school. Option (1) may be used only once during the semester.

Important: Quizzes may be unannounced, and they may sometimes be given in consecutive classes. The lowest quiz or homework grade overall will be dropped. To receive a good grade for the course, it is important to attend nearly every class and to have reviewed your notes from at least the previous week of classes beforehand.

Communication: The primary form of communication within this course is through Canvas. We will make announcements through Canvas, and also provide channels for homework clarifications and team-based communication. As such, we highly recommend that students check Canvas regularly with notifications enabled. The instructor and TAs will also hold weekly office hours to provide support with course materials and projects. You can find the office hours schedule on Canvas as well. You can contact the course instructors and TAs via Canvas and email. Your team will also have assigned TAs as your primary point of contact for additional support and questions.

Computers and devices in class: Research shows that using devices on non-class related activities harms both the device user's learning, and other students' learning. Therefore, in general, we do not allow the use of devices during lectures. If you genuinely use your laptop for class-related activities (note-taking, etc), tell us, and we will make an exception. However, we ask that if you do so, you are careful to keep your devices in note-taking mode (and don't stray to social network, homework, etc). Note that recitation activities will often involve devices, so please bring your laptop!

Project: Students are expected to complete a software development project, whose work products include: **a software requirements specification, a design document, test-case specifications, implementation code, and a brief user's manual.** Teams consisting of 4 students will work together on a development project. In special cases, teams with 3 or 5 students can be approved by the instructor. Projects are proposed by each team and must be approved by the instructor.

Scheduled “live” team **demonstrations** of work products and development practices (e.g., running code, testing, version control, bug tracking, code-coverage measurement) are required. Each team member must contribute significantly to each project work product or deliverable. Each team member’s contribution to each work product should be indicated in a work breakdown, which specifies who was responsible for each task and what percentage of the work they did. If a team member fails to contribute adequately, this should be reported to me promptly by the team leader – don’t wait until the end of the semester!

Teamwork: Teamwork is an essential part of this course. Projects have components that are graded for the entire group and components that are graded individually. Being able to address team issues in software projects is one of the core learning objectives of this class. Guidance on teamwork, reflection, and conflict resolution will be provided throughout the semester and are an essential component of the class. We expect significant efforts in attempting to address the team issues before asking instructors and TAs to step in. However, we will always be available to provide advice on how to navigate these issues.

Academic Integrity Policy: All students in this course are expected to adhere to CWRUs standards of academic integrity. Cheating, plagiarism, misrepresentation, and other forms of academic dishonesty will not be tolerated. This includes but is not limited to, consulting with another person during an exam or quiz or observing their work, turning in written work that was prepared by someone other than you, making minor modifications to the work of someone else and turning it in as your own, or engaging in misrepresentation in seeking a postponement or extension. Ignorance will not be accepted as an excuse. If you are not sure whether something you plan to submit would be considered either cheating or plagiarism, it is your responsibility to ask for clarification. For more information, please go to https://bulletin.case.edu/undergraduate-academics/academic-integrity/.

Use of Generative AI Tools: Generative AI tools such as ChatGPT must not be used during quizzes or tests. For requirements, design and testing documents, AI tools may be used only on the next-to-last version of a document to improve the English, provided this is clearly indicated in the document. Generative AI tools may be used to generate code, provided this is indicated in writing to the instructor and the assigned grader(s) and provided the queries that were used are provided in a text file. However, any use of AI comes with two caveats: (1) you must clearly indicate the use of such technology (and name the tool) in every homework question or source code file where an AI tool was used as a form of attribution, just as you would if you had taken help from a friend; (2) you are liable for factually inaccurate answers or unspecific rambling produced by AI tools; it is your responsibility to edit AI-produced content before submitting it for class purposes.

Disability Resources: In accordance with federal law, if you have a documented disability, you may be eligible to request accommodations from Disability Resources. In order to be considered for accommodations, you must first register with the Disability Resources office. Please Contact Disability Resources by phone at (216) 368-5230, by email at disability@case.edu, and online at https://case.edu/studentlife/disability/, to get more information on how to begin the process. Please keep in mind that accommodations are not retroactive.

Additional resources: Check Canvas. You can check Kevin Smith Library for additional resources and reading materials: https://researchguides.case.edu/cds.

Detailed syllabus (tentative):

Week	**Topic**	**Subtopics**
1	What is Software Engineering?	● Course Overview
1-2	Software Development Process Models	● Waterfall Model ● Rapid Prototyping ● Online Controlled Experiments ● Incremental Development ● Agile Development ● Spiral Model
3-4	Software Requirements	● What are Software Requirements? ● Requirements Elicitation and Analysis ● Specifying Requirements ● Validating Requirements ● Maintaining/managing Requirements ● Inspections and Reviews
5-8	Software Design	● Design Dimensions and Principles ○ Managing Complexity ○ Design Decomposition ○ Designing for Change ○ Encapsulation and Information Hiding ● Design Techniques ○ Stepwise refinement and related techniques ○ Event-Driven Design ○ Object-oriented Design ○ Unified Modeling Language (UML) ● Design Documentation ● Design Tools ● Design Patterns
9-12	Validating Software	● Testing ○ Testing Phases ■ Unit Testing ■ Subsystem Testing ■ Integration Testing ■ System Testing ■ Acceptance Testing ■ Beta Testing ○ Synthetic Testing ■ Specification-based Testing

		■ Code-based Testing ■ Fault-based Testing ■ Interaction Testing ○ Field Testing ○ Regression Testing ● Static Analysis Tools ● Dynamic Analysis ● Statistical Reliability Estimation
13-14	Continuous Integration and Deployment	
13-15	Project Presentations	