CISC 475/675: Software Engineering Principles and Practices

University of Delaware, Spring 2024

Syllabus

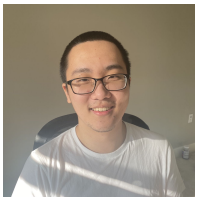
Version of Feb. 4, 2024

All information here is subject to change. Changes will be announced in class and on Canvas.

1. Fundamentals

Instructor: Stephen Siegel, `siegel@udel.edu`. Office hours: Mon/Wed 10:10–11:00 AM, or by appointment, Smith Hall 432.

Teaching Assistant: Alex Wilton, `awilton@udel.edu`. Office hours: Fri. 10:00–11:00 AM, Smith Hall 102A, or by appointment.



Teaching Assistant: Yufan Wang, `leeff@udel.edu`. Office hours: Tue. 2:00–3:00 PM, Smith Hall 102A, or by appointment.

Class meetings: Alison Hall, Room 222, 8:40–10:00 AM, Mon/Wed from Mon. Feb. 5 to Wed. May 15, except Mon. Mar. 25, and Wed. Mar. 27. Total number of classes: 28.

Midterm exam: In class, Wed. Mar. 20.

Final exam: Alison Hall 222, Monday, May 20, 11:30 AM–1:30 PM.

Texts: There is no required text. We will instead read individual articles that will be provided to you at no cost. I will also be recommending books on various topics throughout the semester.

Course Canvas page: `https://udel.instructure.com/courses/1763122`. The name of the site is `24S-CISC475-010`. Note that the same site is used for CISC475 and CISC675 students. Canvas will be used primarily to share documents, to submit homework, and for grades.

Slack workspace: `https://se-ud.slack.com`. To sign up, go to `https://join.slack.com/t/se-ud/signup`. We will use Slack for asynchronous discussion of material and to ask and answer questions. Do not use Slack for discussion involving any confidential information, such as your grades or academic record—for these things, use email, phone, or Zoom conference.

2. Course Abstract

This course is an advanced introduction to the major themes of software engineering practice and research. Topics include: software engineering processes and methodologies; requirements analysis and specification; software design principles, patterns, and formalisms; object-oriented design;

implementation, documentation, and traceability; unit, integration, and systems-level testing; coverage metrics; test-case development strategies; test automation; software verification and formal methods.

We will explore these topics by reading and discussing many of the landmark articles in the software engineering literature. These include some of the classic contributions from foundational figures, such as Fred Brooks, Mary Shaw, David Parnas, Michael Jackson, and Barry Boehm. We will also sample newer, cutting-edge work, such as Daniel Jackson's work on *concepts* and the Alloy Analyzer.

Throughout, the focus is on foundations and underlying principles, ideas that are time-tested and will remain relevant for the foreseeable future. This is more valuable than learning the latest technology or fad.

3. How this Class Will Operate

Reading is the key to success in this course. There will generally be a reading assignment for each class. The details will be available on Canvas. For each reading, there will be a timed Canvas quiz which must be submitted before class.

Each class will typically consist of two parts. Class will begin with discussion of the reading assignment due that day. Students will work in small groups of up to 6 people, who are sitting at the same table. There will be 1–3 questions I will ask you to discuss, based on the reading. The group will choose one member to be the *scribe*, who will take notes about the discussion on a shared Google Doc. The group will also choose one person to lead the discussion of each question. After a prescribed amount of time discussing each question, the class will reconvene and question leaders will summarize the conclusions reached by their groups. Discussion among the whole class will encourage further understanding.

The second half of the class will usually consist of a lecture. The topics will generally complement the reading, go into certain areas in more detail, or provide some background for whatever is coming next.

There will also be 5 homework assignments. Each assignment will be given out at least 2 weeks before it is due. Students are encouraged to begin assignments early so that they can attend office hours if needed.

There will be a midterm and a final exam. One week prior to each exam, a list of sample problems will be distributed. The problems on the exam will be some subset of the problems from the list, with small variations. There should be no surprises on the exam.

Students registered for 675 will get different assignments and exams than those registered for 475. The 675 versions will be more challenging. The schedule is the same for everyone.

The Slack workspace may be used to ask and answer questions, post interesting thoughts, or discuss anything related to this class or software engineering generally. It is recommended that everyone install the Slack tool (or use it through the web browser) and check the workspace regularly. Important announcements will also be posted on Canvas.

4. Grading

Quizzes: 20%. Homework: 25%. Midterm: 20%. Final: 25%. Participation: 10%.

Participation grade is based primarily on showing up and contributing to the discussions of the readings, but also on asking or answering questions (including electronically or in office hours), and submitting the final course evaluation. Attendance will be taken at each class.

Note: if you are *sick*, even if you think it is just a cold, don't come to class. You must email Prof. Siegel and cc both TAs before class, for your attendance to be excused. For the midterm and final

exams, you will show up if you are experiencing minor illness. Wearing a mask is encouraged if you are experiencing respiratory symptoms. In the event that you are too sick to take the exam, you will email Prof. Siegel (and cc both TAs) at least 12 hours beforehand.

Late policy for Homework: The deadline for HW is the beginning of class (i.e., 8:40 AM sharp). If you turn in HW after deadline but before start of next class, it will be graded with 10% penalty. Everyone gets one one-class-late submission without penalty. Late homework cannot be accepted for the assignments due just before the midterm or the final exam, because solutions will be released on the due date.

Letter grades are obtained from the numerical score as follows:

Minimum score	93	90	87	83	80	77	73	70	67	63	60	0
Letter Grade	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F

5. Academic honesty

If you think it will help, unless stated otherwise, you may use Large Language Models such as Chat-GPT and the like to assist on your homework, as long as you clearly explain in your submission how you used the tool (give exact queries and responses). Keep in mind that Chat-GPT frequently *plagiarizes*—it will present someone else's (often copyrighted) work without telling you where it came from. If you turn in plagiarized work, you are guilty of plagiarism, regardless of whether you got the material from Chat-GPT or copied the original material directly.

You are also free to work on the homework with other students in this class. However, you should not just copy someone else's solution; it should be a collaborative effort. If course staff suspect you are abusing this privilege, they will speak with you about it. Do *not* discuss the homework with anyone who is not registered for this course.

Quizzes are to be taken alone. Do not communicate with anyone about the quiz (other than the instructor and TAs) until the deadline has passed. You should also not use LLMs or search tools. The only material you need is the assigned reading.

Plagiarism. Copying any other person's work (off the Internet, for example) without proper acknowledgment is *plagiarism*, one of the most serious academic offenses. This will result in charges being filed in accord with the University's Policy on Academic Honesty. The same holds for using LLMs such as Chat-GPT without precise attribution. Avoiding plagiarism is not difficult: just be scrupulous and precise in writing down any sources you used for any idea, solution, quote, etc. Students are encouraged to ask course staff if they need further clarification.

6. Class Rules

(1) **No cell phones in class.** They are major distraction devices. All cell phones must be turned completely off and put away so that they are not visible. Violations will result in negative participation points.
(2) **Be kind.** Treat everyone with respect: students, instructor, and TAs. Handle disagreements civilly. If anyone is giving you problems, speak with the instructor as soon as possible.
(3) No chewing gum in class.
(4) In the class room, computers are to be used only for class activities. Not for social media, browsing the web, or anything else.

7. Schedule

The schedule is provisional. Adapting to change is a major theme of software engineering and the same is true in this class. Things may move slower, or faster, than planned, or the interests of the students may dictate changes of course. In any case, any change will be announced with plenty of lead time.

	Date	HW	Lecture	Discussion	File
1	Mon, Feb 5, 2024	**Introduction: Class mechanics, History of Software Engineering, Core SE Principles**			
2	Wed, Feb 7, 2024		Principles, cont.	Frederick P. Brooks, Jr., **No Silver Bullet: Essence and Accidents of Software Engineering**. Computer 20 (4). IEEE. April 1987. [https://doi.org/10.1109/MC.1987.1663532]	`brooks~1987~silverBullet-WEB.pdf`
3	Mon, Feb 12, 2024		Math background: Set and Relation Theory	Mary Shaw, **Prospects for an Engineering Discipline of Software,** IEEE Software. Nov. 1990	`shaw~1990~engineering.pdf`
4	Wed, Feb 14, 2024	HW1 Release: sets, relations, ERD, DFD, requirements	Math background, cont.: one, many, transitive closure, hierarchies, levels	W. W. Royce, **Managing the development of large software systems: concepts and techniques**. ICSE '87: Pages 328–338, reprinted from 1970.	`royce~1970~waterfall.pdf`
5	Mon, Feb 19, 2024		Entity Relationship Diagrams	Bertrand Meyer, **Handbook of Requirements and Business Analysis.** Springer 2021. Chapter 1: Requirements: basic concepts and definitions, pp. 1–18	`meyer~2022~requirements-book.pdf`
6	Wed, Feb 21, 2024		Data Flow Diagrams	B. W. Boehm, **A Spiral Model of Software Development and Enhancement**, Computer, IEEE. May 1988.	`boehm~1988~spiral.pdf`
7	Mon, Feb 26, 2024		Agile Processes	P. Deemer, G. Benefield, C. Larman, B. Vodde, **The Scrum Primer: A lightweight guide to the theory and practice of Scrum, Version 2.0**. 2012 [http://www.scrumprimer.com]	`deemer-etal~2012~scrumprimer20.pdf`
8	Wed, Feb 28, 2024	HW1 Due	Requirements	Bertrand Meyer, **Agile! The Good, the Hype, and the Ugly**. Springer, 2014. Chapter 1: Overview	`meyer~2014~agile-overview.pdf`
9	Mon, Mar 4, 2024	HW2 Out: Processes, Alloy 1	Concepts	Daniel Jackson, **Three Stages of Enlightenment: A brief intro to concept design in three parts**. May 2, 2023, [https://essenceofsoftware.com/posts/three-stages/]	`jackson~2023~three-stages.pdf`
10	Wed, Mar 6, 2024		Alloy	Michael Jackson and Pamela Zave, **Deriving Specifications from Requirements**, ICSE 1995, pp. 15–24	`jackson-zave~1995~spec-example.pdf`
11	Mon, Mar 11, 2024		Alloy	Daniel Jackson, **Alloy: A Language and Tool for Exploring Software Designs**. Communications of the ACM. Sep. 2019, 62(9). DOI:10.1145/3338843	`jackson~2019~alloy.pdf`
12	Wed, Mar 13, 2024		Alloy	Alloy Tutorial I, https://haslab.github.io/formal-software-design/	
13	Mon, Mar 18, 2024	HW2 Due **(strict deadline)**	Alloy	Alloy Tutorial II	
14	Wed, Mar 20, 2024	**MIDTERM EXAM**			
15	Mon, Apr 1, 2024		Alloy. Introduce module theory.	Alloy Tutorial III	
16	Wed, Apr 3, 2024	HW3 Out: Alloy 2, Module Design	Alloy	Alloy Tutorial IV	
17	Mon, Apr 8, 2024		Module Theory	D. L. Parnas, **On the criteria to be used in decomposing systems into modules.** Comm. ACM, 15 (12), Dec. 1972, 1053--1058.	`parnas~1972~decomposing.pdf`
18	Wed, Apr 10, 2024		Module Theory	D. L. Parnas, P. C. Clements, D. M. Weiss, **The Modular Structure of Complex Systems.** IEEE Trans. Soft. Eng., SE-11 (3), March 1985	`parnas-clements-weiss~1985~modular.pdf`
19	Mon, Apr 15, 2024		Patterns	D. L. Parnas, **Designing Software for Ease of Extension and Contraction**. IEEE Transactions on Software Engineering, SE-5(2) March 1979	`parnas~1979~extend-contract.pdf`
20	Wed, Apr 17, 2024	HW3 Due HW4 Out: Patterns, Testing	Patterns. Introduce testing	Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, **Design Patterns: Abstraction and Reuse of Object-Oriented Design**. ECOOP '93, LNCS 707, pp. 406–431, 1993.	`gamma-etal~1993~design-patterns.pdf`
21	Mon, Apr 22, 2024		Testing	G. Fraser and A. Arcuri, **Whole Test Suite Generation.** IEEE Transactions on Software Engineering 39(2), pp. 276–291, 2013.	`fraser-arcuri~2013~evosuite.pdf`
22	Wed, Apr 24, 2024		Testing. Introduce symbolic execution.	Amy J. Ko and Brad A. Myers, **Debugging Reinvented: Asking and Answering Why and Why Not Questions about Program Behavior**. ICSE 2008, ACM.	`ko-myers~2018~debugging.pdf`
23	Mon, Apr 29, 2024		Symbolic Execution. Introduce KLEE.	James C. King, **Symbolic Execution and Program Testing.** Comm. ACM 19(7), July 1976.	`king~1976~symbolic.pdf`
24	Wed, May 1, 2024	HW4 Due HW5 Out: Symbolic Execution, find bugs, contracts	Symbolic Execution. CIVL. Introduce find-bugs.	Cristian Cadar, Daniel Dunbar, and Dawson Engler, **KLEE: unassisted and automatic generation of high-coverage tests for complex systems programs**. In Proceedings of the 8th USENIX conference on Operating systems design and implementation (OSDI'08), 2008. USENIX Association, Berkeley, CA, USA, 209–224.	`cadar-etal~2008~klee.pdf`
25	Mon, May 6, 2024		Static Analysis. Introduce Hoare Logic.	David Hovemeyer and William Pugh. 2004. **Finding bugs is easy**. In Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications (OOPSLA '04). ACM, New York, NY, USA, 132–136. DOI:10.1145/1028664.1028717	`hovemeyer-pugh~2004~findbugs.pdf`
26	Wed, May 8, 2024		Hoare Logic	C. A. R. Hoare, **An Axiomatic Basis for Computer Programming**. Comm. ACM, 12(10), Oct. 1969, p. 576--580,583.	`hoare~1969~axiom.pdf`
27	Mon, May 13, 2024		Contracts	B. Meyer, **Applying "Design by Contract"**. Computer. Oct. 1992. IEEE. pp 40–51	`meyer~1992~contract.pdf`
28	Wed, May 15, 2024	HW5 Due **(strict deadline)**	Contracts Frama-C	Patrick Baudin, François Bobot, David Bühler, Loïc Correnson, Florent Kirchner, Nikolai Kosmatov, André Maroneze, Valentin Perrelle, Virgile Prevosto, Julien Signoles, and Nicky Williams, **The Dogged Pursuit of Bug-Free C Programs: The Frama-C Software Analysis Platform**. Communications of the ACM, August 2021 64(8), pp. 56--68.	`baudin-etal~2021~frama-c.pdf`
	Mon, May 20, 2024	**FINAL EXAM**			