CS 684 (Sp24)

Syllabus

Welcome to CS 684! This class is focused on software quality: both how to assess it and how to achieve it in your own work. The course therefore has two technical focuses: **testing** and **static analysis**, both of which are important for assuring quality in the real world. We'll spend roughly 60% of the course (through spring break) on testing, and the rest of the course on static analysis. My goal in this course is to get you up-to-date with the best practices at the **very best** software engineering firms in the world in these two topics: that is, the goal of this course is to give you a firm foundation in the "state-of-the-practice" in industry. Note that reaching this level requires us to engage with the research literature, as the top engineers at the best software firms do—– especially the most important papers from the past 10-20 years.

Prerequisites

While this course has no official prerequisites, I will assume in this course that you already know how to program: that is, that if I tell you to go write some code, you'll be able to go do it. Since this course focuses on how to program **well** (i.e., how to engineer software!), you first need to know how to program at all. I'll also assume some familiarity with command line tools, debugging, and using a search engine: I expect that if I ask you to go write code in some language you've never seen before, you'll be able to find the necessary components online, find an online tutorial on the syntax, and figure out how to write that code. Put another way, I won't teach you how to write a program: this course already assumes that you can do that. It is a graduate-level CS elective, after all.

Topics

The course is primarily structured around covering two topics in depth: testing and static analysis. We will cover testing in the greatest depth, including at least the following topics:

- formal definitions of testing
- basic testing terminology and theory (e.g., unit vs integration testing, regression tests, etc.)
- coverage

- fuzzing and random test input generation
- mutation testing and analysis
- oracles and oracle generation
- differential testing

In addition, we will cover other analysis techniques for quality assurance. These will include at least:

- dataflow analysis
- abstract interpretation
- reduction of program analysis problems to SMT
- other useful dynamic analyses beyond testing, such as dynamic race detectors

Course Structure

This is an evening class in a three-hour block. Three hours is a long time, and I do not think you all want to listen to me talk for that long; further, I do not want to lecture for three hours straight every week. With that in mind, almost all classes will be split into two parts: a "lecture" part first, followed by a in-class exercise. Each in-class exercise is also a homework assignment (and all homework assignments begin as in-class exercises). These assignments are generally due one week later (i.e., by the end of the day before the next class), though there is one exception to this (the mutation testing assignment is longer and worth more points, so I've spread it over two weeks; there is no in-class for "other dynamic analyses" to compensate for that). During the in-class exercise period, the TA and I will circulate through the room and answer questions about the assignments.

Grading and Assignments

Your grade is composed of the following sub-scores (in no particular order):

- 60%: Homeworks
- 25%: Exams (10% for the midterm, 15% for the final)
- 15%: Participation & Professionalism

This class will be curved: when grading, I prefer to use the whole range available rather than scores in a tight range. That is, if an assignment is worth 10 points, I will give grades at all the

points between 0 and 10. I will project your raw scores onto the final distribution twice during the semester:

- after the mid-term exam
- shortly before the final exam

You will be notified of your current projected class grade via email at each of these points.

Readings and Reading Responses

Each lecture has mandatory readings. I expect you to read mandatory readings before coming to class that day, and I will check that you've done so using reading quizzes (see Participation & Professionalism, below).

Participation & Professionalism

Your participation & professionalism grade is composed of two scores.

First, your *Professionalism* score is based on the instructors' impression of how well you participated in class, with deductions for distracting other students and credit for asking and answering questions (either in person or on the course discussion board).

Second, your *Participation* score is based on reading quizzes at the beginning of most lectures. You get half credit on these quizzes just for being there, and half credit for answering the reading questions correctly (the questions will always be easy if you did the reading). For full participation, you need to get at least a score of 70% on all quizzes over the whole semester (this gives you space to e.g., miss a reading quiz because you were sick or have a family emergency - there are no excuses for missing reading quizzes). Put another way, you can miss up to 30% of the reading quiz points and still get full participation points.

These policies are designed to encourage you to come to class. A big part of the goal of this class is to help you develop an intuition for what good software engineering looks like, and without coming to class you won't get the full benefit of that intuition.

Remote Participation

Generally this class does not support remote participation: teaching is much more effective, in my experience, when everyone is physically present. However, I understand that sometimes you are sick, traveling, or otherwise unable to come to class. I will arrange for remote participation in any particular lecture as long as you request it at least one hour in advance (if you're sick or in some other emergency) or 24 hours in advance (if you're traveling or otherwise planning to be unable to

come to class). Notify the instructor via email if you need to participate in a particular class remotely.

Asking Questions

There is a course Discord server which you can use to ask (and answer) questions about any of the course topics or for help with the homework. Participating on Discord is optional, but if you do participate in a productive manner (especially by answering other student's questions!), it can have a positive impact on your participation score.

Exams

There are two exams in this course:

- a mid-term, which is held in class about halfway through the semester (worth 10% of your course grade)
- a final exam, which is held during the university-scheduled final exam slot (worth 15% of your course grade)

Both exams will cover a range of topics discussed in lecture and/or in the mandatory readings, from any time during the semester up to the point when the exam is held. The exam will be comprehensive, covering many of the topics we discuss; I may ask about anything we covered in class or that you were supposed to read. The exam will be conducted in person. Contact the course staff privately via email if you are not able to attend for any reason (e.g., you are sick or need special accommodations) and we will arrange an alternative. See the exams page for more information.

Collaboration Policy

Collaboration is generally encouraged in this course, as is consulting online resources. You are permitted to copy small amounts of code from any source except another student's copy of an assignment, *as long as you cite your source*. "Another student's copy of an assignment" also includes students not currently enrolled in the course - e.g., students who took this class in previous semesters or took classes that used similar materials at other institutions. To make this more clear, here are some examples of acceptable and unacceptable collaboration on a programming assignment in this course:

Acceptable collaborations:

- Discuss problems/solutions/anything with any number of other students (as long as you don't look at each other's code).
- Copy a short (about 10 lines or fewer - use your judgment) snippet from stackoverflow.com or a similar source, as long as you include a comment with the source URL.
- Copy code written by one of your teammates during the group project for another part of the group project.
- Copy code from the output of a large language model such as ChatGPT that you prompted yourself, if you include a link to a record of your interaction with the model (e.g., ChatGPT's "share" feature) as a code comment.

Unacceptable collaborations:

- Copy code directly from another student on an individual project.
- Copy code from another group on a group project.
- Copy a significant portion (more than about 10 lines of code or a single method - use your judgment) of your assignment from the internet, even if you cite your source.
- Copy a short snippet from the internet without citing your source.
- Copy code from the output of a large language model (such as ChatGPT) without citing your source
- Copy code from the output of a large language model prompted by someone other than you

These rules are intended to mimic what is acceptable in industry when working as a software engineer: using the resources available to you, such as your teammates and the wider internet, is always allowed. But, it would be illegal to copy code from a competing company working on a similar product. The same rules apply to English writing, although you should avoid copying text whole-sale from any source (this specifically means that using an LLM to help you with writing is allowed, but do note that I don't think LLMs produce particularly lucid text, so you should definitely avoid using LLM-generated text without editing it heavily if you want to do well on a written assignment).

Consequences of Violating the Collaboration Policy

(From the University)

"Academic Integrity is the cornerstone of higher education and is central to the ideals of this course and the university. Cheating is strictly prohibited and devalues the degree that you are working on. As a member of the NJIT community, it is your responsibility to protect your educational investment by knowing and following the academic code of integrity policy that is found at: http://www5.njit.edu/policies/sites/policies/files/academic-integrity-code.pdf.

Please note that it is my professional obligation and responsibility to report any academic misconduct to the Dean of Students Office. Any student found in violation of the code by cheating, plagiarizing or using any online software inappropriately will result in disciplinary action. This may include a failing grade of F, and/or suspension or dismissal from the university. If you have any questions about the code of Academic Integrity, please contact the Dean of Students Office at dos@njit.edu"

Late Policy

You may use a late day on a homework assignment up to twice throughout the semester. Assignments are generally due on a particular day "anywhere on Earth" (AoE), which is typically 7am the following day on the US east coast. Using a late day allows you to submit up to 24 hours later without penalty. Assignments turned in more than one day late or after you have already used two late days during the semester will not be accepted.

Acknowledgments

This course is heavily inspired by a number of other courses in software engineering at other universities, especially:

- René Just's CSE 504P at the University of Washington
- Michael Ernst's CSE 503 at the University of Washington
- Wes Weimer's EECS 481 at the University of Michigan

Thanks especially to René and Wes for permitting me to reuse assignment specifications from their courses.

As a student, if you're looking for more materials (or just a different perspective) on any of the topics we cover, you might start with those (excellent) courses.