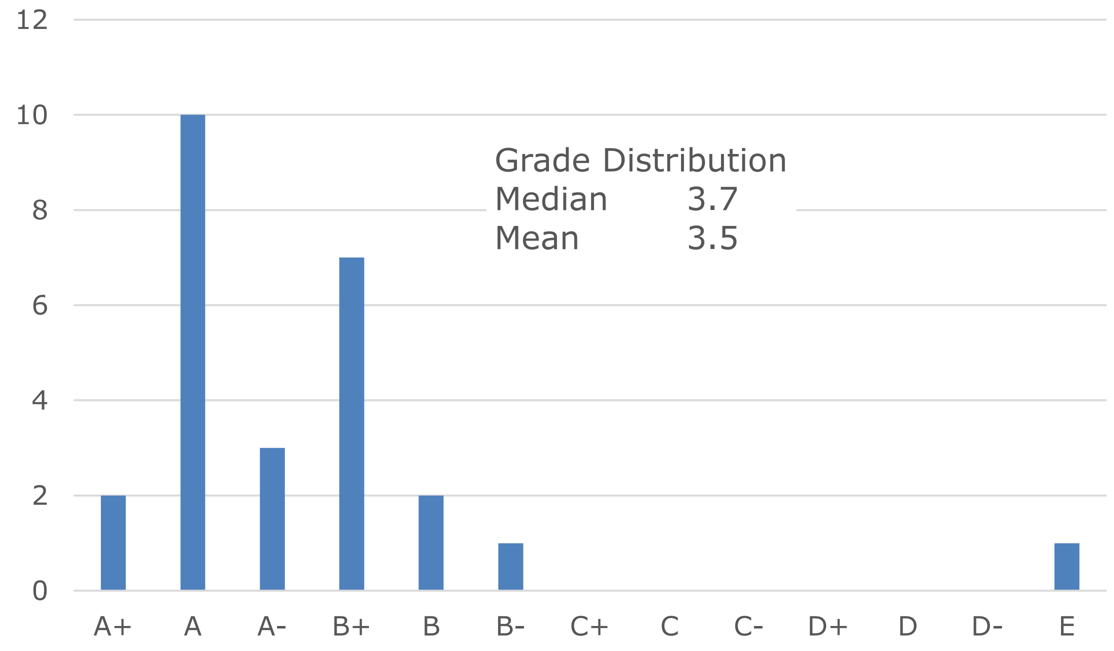
EECS 398 System Design in C++
Winter 2018

Final grades and reflections[1]

I've posted grades for all of you to Wolverine. Here was the distribution. I had promised a minimum 3.4 median but ended up giving you one that was considerably easier.



Your group score was based on how you ranked compared to other teams, 60% project presentation, report and final review, 20% group photo plus 10% each for initial project plan and your hash table homework. I intended the milestones as coaching sessions and did not include them in grading.

Your individual score was 30% your group score, 30% your individual contribution to your team, 15% each on midterm and final, and 10% homework 1.

I did read all of your final essays at least four times, all of them twice before marking them up with comments, then again one more time to rank them into tiers and assign grades.

Observations

Median size of the engines was 5300 LOC in C++, high was 14,271, low 4170. Individual contributions ranged from zero to 4056 LOC; median was 1006 LOC. My actual grading

[1] Originally posted to the class via Canvas. This version has some minor redactions and corrections.

considered much more than just LOC, but the way it worked out, a student who contributed 500 lines to their team got a B, if they contributed 1100 LOC, they got an A.

In my first lecture, I mentioned that the group photo assignment was often a strong predictor of future performance. That turned out to be true. C++Lue submitted both the winning group photo and the winning engine and they've given me permission to post their winning paper on Canvas so you can see how they did it. Realistically, they crushed every other team except BAKAN, which beat them on index size with an impressive 13.4M documents thanks to Nathan's 4-day crawl.

C++Lue worked exceptionally well as a team, took advantage of individual team member's skills, wrote a LOT of code, and showed initiative and creativity to go well beyond what was asked, e.g., training using data scraped from Google, considering politeness in their crawl, carefully constructing of a seed list of URLs to start the crawl, and using the tiny web server I gave you to create a web front end.

Student evals

I've also posted the results of your student evaluations to Canvas. Complaints that I need more content, not just slides with a lot of code, and that if I'm not going to require 482, I need to do a much better job on the OS topics, especially multi-threading, and, frankly, many others, are completely valid. But it's still brutal to read some of this.

Background

When I arrived last fall, the plan was that I would teach David Kieras's 381. But I wasn't a real good fit for that and also, I wanted to swap in a search engine project (which had the advantage of being something I knew how to do) and make other changes.   Since Dr. Kieras will be teaching 381 exactly as he has in the past, the decision was taken to make this a new experimental class. I had about four days last December to write and submit the syllabus and the design project documents to the curriculum committee if it was going to make it into this semester.

My objectives

I wanted it to feel as much as possible like a real dev job on a team building a new product from scratch that you don't already know how to build. I wanted the project to feel significant and meaningful, the sort of thing you could talk about with recruiters and your friends or family and they could understand what you'd done.

I also wanted it to be accessible, even to students who only had 280 and 281, even though that meant I would have to somehow explain the filesystem, threads, locks and sockets to students who had likely never seen any of it before. But that also seemed like one of the major points of system design, that it's typically more tightly coupled to the OS. It's "closer to the metal".

There's also a lot more *ambiguity*, which nearly all of you commented on. You get more freedom to do whatever you want, but there's also the possibility of suffering the consequences of a poor decision.

I've been fortunate that all the projects I've worked on my entire career have always been new ("clean sheet of paper") designs. The first line of code (at least in my piece) was always yet to be written and it was always mine. When you start there, it's always the same, especially if you're joining a team: You're adrift, with little idea where to begin, feeling incompetent. Then you begin to figure it out. The payoff is that what you write is *yours,* you've *created* it and *it's just the way you want it.* You get lots of chances to be creative, do research and experiments, to think about what's unique or special about the problem in front of you and how you might decide to solve it with a completely original design.

Because of the ambiguity and the need to make lots of design choices that could be either really great or really awful, I think system design depends more than perhaps other areas in computer science on *intuition* and *strategy* about how to build stuff, stuff that's bigger than you've done before, stuff that needs to be broken into pieces written by different people that have to talk to each other and will determine what the customer gets. I think a lot of this is learn by doing.

I get a lot of satisfaction when it's turned out I've made a good choice, when I feel like I had a good insight into how to approach a big problem and it worked. But I wouldn't always know right away. On big projects, I've often had to keep writing and writing and writing, sometimes for months at a time, trusting that I had proved to myself that my strategy was right and my design would work. But it often was just trust that it would work. Sometimes with a difficult piece of code, especially if there's lots of recursion, you can "prove" to yourself that something should work, but you still need to see it to believe it. So, when something does finally wake up, and I find I was right, and it works, I can sometimes bounce off the walls.

I also feel good when I think I've done something kind of hard but in an elegant way, and my code is just pretty and simple and clean. If someone were to read it, I want them to decide, *"Well, if I'd known it was that easy, anyone could do it."*

Reflections

Your feedback on the student evals notwithstanding, I'm declaring success.

All 5 teams successfully delivered working search engines. Yippee! That really made me proud of you. I also felt personally relieved as well that my basic plan, a whole search engine from scratch in C++ by a team of 5 students had worked, that the whole project as I'd defined it (or hadn't :) really was doable and that I could fill in the technical pieces you wouldn't already know from 280 and 281 well enough you could all be successful.

For most of you, this was the biggest project you've done. One of you told me in your essay about having trouble getting interviews but now, a simple mention of writing a whole search engine instantly draws invites. That's exactly what I hoped. I wanted it to be a big meaningful, relatable design project you could talk about it. If you got that and a sense of the ambiguity and the need to make choices that you see on a real project, I'm pretty happy.

I totally get that that this course was very raw and detracted from the experience for many of you. I'll be giving it again next winter 2019 and have some development funding over the summer to build more content, more slides, more homework, more everything. And with some veterans from this first run, I'll be able to hire IAs next winter to help me run the course and make their own contributions to the content. It won't just be me next time and it won't be just whatever minimal slides and codes samples I could create from scratch every two days. (The code samples were over 4200 lines.)

Beyond the need for additional course content across the board, I can see that I need to pay more attention to the OS topics, especially multi-threading, which many students seeing it for the first time reported as bewildering. I want this course to be accessible to students who've only have 280 and 281, giving students a chance to do a big project early in their careers, rather than making them wait until they're graduating seniors. So rather than requiring 482, the OS class, I'm determined to do a much better job on the OS topics and warn incoming students that if they've already taken 482, it could be repetitive.

Finally, I got some complaints in the essays and in the evals that the class wasn't about system design and that they'd expected design patterns, didn't realize the class would be one big project and that they'd been hoping for more general design principles. Some claimed they learned nothing.

I'm sorry to have disappointed you in other ways, but I don't think I can own these particular complaints. The syllabus made no mention of design patterns and promised the C++ topics would already be familiar from 280. I tried very hard to make clear in the first paragraph that the major effort would be a design project building a search engine from scratch. But also, I think system design is often about deciding what is unique about the problem and using that to produce a better solution.

Most of all, I think you did learn how to work on a team to design a system you didn't already know how to do, which is all I really promised. And I point to you and your engines as proof I delivered. You did it. All of your engines did work. Congratulations. Now, go do it all over again on your next big project you don't know how to do.

I wish you all great success.

Nicole