CS61B, 2021

Lecture 37: Software Engineering IV

- Software Engineering Engineering

61B: But One of Many Possibilties

We're now mostly done with 61B!

Today, as an experiment, we'll be reflecting on 61B itself.

- How was it designed?
- What did people think?
- What works about the class? And what doesn't?
- What is the broader system in which 61B is embedded?
- What else could 61B (and all education) be?

61B Design Constraints

As a professor, I have more or less total control over the course.

- We could have no exams.
- We could have no labs.
- We could have no theory assignments.
- We could have no deadlines for assignments.
- We could have no programming assignments where we actually run code.
  - Dijkstra was a famous proponent of this approach (see end of this).
- All assignments could be partner/group assignments, or none of them.
- Exams could involve partners (have you done this in a class you've taken).
- It could be taught in C++, Kotlin, Python, OCaml, etc.

*Technically, I'd have to go through COCI to make some of these changes.

Questions About Course Design?

- Example: “Why 6 slip days? I heard they had 15 last semester?”
- Why Gitlet, not BearMaps? [coming later]
- Why Ed? The Fall 2020 course staff voted for Ed. I think also the students did.
- Why don’t we have to implement hash tables and binary search trees totally from scratch in the same way we did deques?
  - This has obvious synergy with exam studying.
  - In Fa20 (and some other semesters), we had implement a KdTree, and implement A*, and implement a certain kind of PQ.
  - Idea: Projects and HWs we want you to do something totally new.
  - Idea 2: It’s more exciting to build a system that does something interesting.
  - Idea 3: More room to run into systems design issues (e.g. Gitlet byow)

Questions About Course Design?

- Data Structures: Discrete math as a prereq.
  - Other classes at Berkeley like 70 has graphs.
  - It's hard for me to prove correctness of anything inductively. Asymptotics are weird to do fully rigorously without recurrence relations (sometimes).
  - 61B lets you see all this material

Other 61Bs

61B has been taught since ~Spring 1994.

- Before that, it was CS60C, which goes back to at least 1988.

In modern times there have been 4 varieties of 61B:

- Hilfinger: 4 extremely long real world projects that are somewhat based on data structures material.
- Hug (sp19-fa20): 1 (or 2) long real world project that is somewhat based on data structures material. Remaining material ties in tightly to lectures.
- 61BL: Lab based class that focuses heavily on data structures, but with one large real world project (Gitlet).
- Shewchuk / Yelick (extinct): Focus on implementing data structures. No large real world project.

61B Versions 1.0, 2.0, 3.0

61B 1.0

Gitlet was first offered in the Spring 2015 offering of 61B.

- My first solo offering of the class.
- Projects had significant authorship from students.
  - Project 0 - Bomb Checkers (me, but implemented by Jimmy Lee).
  - Project 1 - ngordnet (me).
  - Project 2 - Gitlet (Joey Moghadam).
  - Project 3 - Fun with Tries (me, adapted from my old Princeton HWs).
- Joey also used the project as one of his assignments in Summer 2015.

Spring/summer 2015 Gitlet was **way too hard**.

- No testing provided.
- No tips on persistence.

61B 2.0

CS61B Version 2 (Spring 2016, 2017)

- Fall 2014/Spring 2015 observation: Hated that students had to split time between the core data structures content and a huge project that wasn't related to that content.
- Decided to have the messy real world project due right before data structures:
  - 2016: Build a text editor.
  - 2017: Create software for manipulating text databases.

The Roundtable

Held a roundtable with 12 students from each quartile to discuss their experience with The Project. Roughly:

- Top quartile: Mostly loved it, no big deal.
- Second quartile: It was really hard, but very rewarding.
- Third quartile: Spent every waking hour thinking about it, managed to just barely make it. Felt amazing when done, though totally exhausted.
- Fourth quartile: Absolutely miserable experience, generally didn't finish. Never felt like they made progress.

Decided to tone down The Project for 61B 3.0.

CS61B 3.0

CS61B Version 3 (Spring 2018, Spring 2019, Fall 2020)

- Proj2: Highly scaffolded project called Bearmaps. Tied into course content tightly.
  - Implement a Kd-Tree. No public autograder.
  - Design and implement your own priority queue with an extra changePriority operation. No public autograder.
  - Use the PQ to implement A*.
  - Use the KdTree and A* to implement a Google Maps like application.
- Proj3: Build your own world.
  - Open ended.

In Fall 2020, TAs felt that projects didn't give students enough independence. Project 2 was basically just "implement these ideas from lecture."

CS61B 3.5

CS61B Version 3.5 (Spring 2021)

- Proj 0: 2048
  - Harder algorithmic design challenge than old project 0.
- Proj 2: Gitlet
  - Much bigger, open ended project than BearMaps.
  - Lots of room for creativity.
  - Tons of existing support resources and a team of TAs who knew the project well.
  - We'll reflect more on it shortly.

List of Changes from Each Semester (in case you're curious)

- Sp15 reflections: Link
- Sp16 reflections: Link
- Sp17 reflections: Link
- Sp18 reflections: Link
- Sp19 reflections: Link
- Fa20 reflections: Link

Reflection on Gitlet

Gitlet: Origins

Spring/summer 2015 Gitlet was **way too hard**.

- No testing provided.
- No tips on persistence.

The Return of Gitlet

After Summer 2015, Gitlet laid dormant.

In Fall 2017: Paul Hilfinger tried out Gitlet.

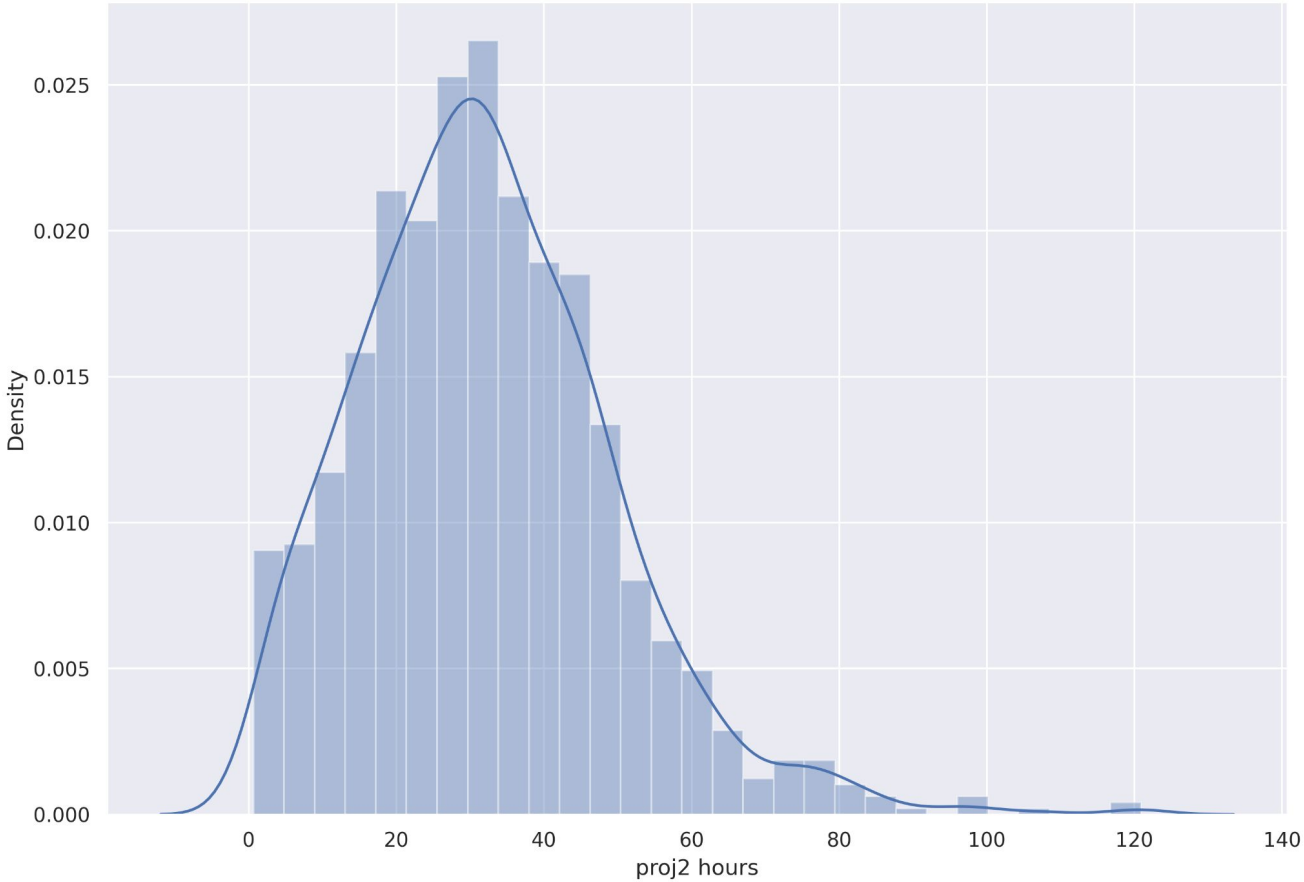
- Has been offered every Summer and almost every Hilfinger semester since.

Gitlet This Semester

In response to significant TA concern that scaffolded BearMaps was too “paint by numbers”, I started revamping the sp15 Ngordnet.

- Insufficient time to get a project we felt good about.
- Gitlet had been so heavily supported...

Time Spent Programming on Gitlet



Only includes time in Intellij!

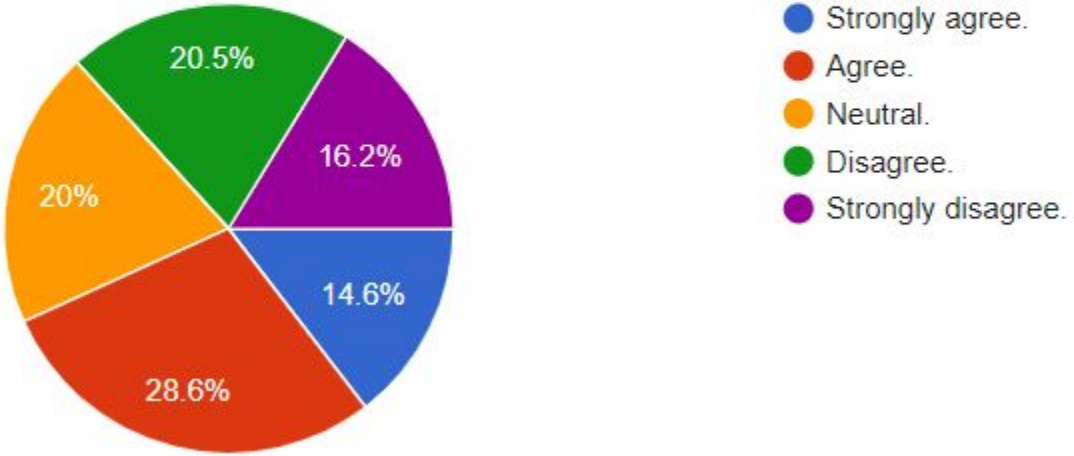
Statistics:

- Mean: 32 hours.
- Median: 31 hours.
- 25%: 20 hours.
- 75%: 42 hours.

Gitlet Comments

Was Gitlet a healthy experience for you?

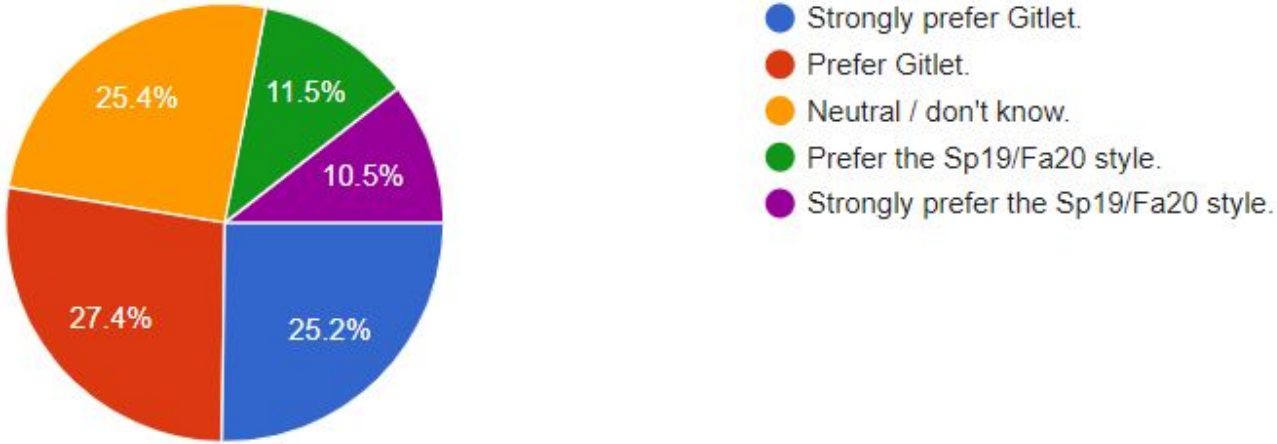
1,194 responses



Gitlet Comments

In Sp19 and Fa20, project 2 was a project which, roughly speaking, was like a double length version of the Deques project. That is, there were specific classes and methods to implement, but not much design. If you had the choice to go back in time and swap the project with the old Sp19/Fa20 style so that you could do that instead, which would you prefer?

1,219 responses



Some Gitlet Comments

“this project crushed my entire soul i hate it. bUT now that it is over it was such a good learning experience omg like what the heCK WAS THAT omg”

“I am sad and I have to switch majors. I was in love with cs but now I hate it. I guess everybody changes, but it was sad for me to let go. Goodbye coding :(“

“While it was a horrible overall experience, I am grateful that I did (half) of Gitlet since I am now confident that I never want to work in SWE. I can't believe people spend more time looking up how to do/use random crap in Java than actually thinking. Wild.”

“I just have never felt so stupid, but once I got it I felt good. I don't think that's healthy, but that's the truth”

Some Gitlet Comments

“I think this was an amazing project. I do think that when you first start it there are many students who would be too intimidated to start or may set them up for failure because the increase in difficulty from proj1 to proj2 was insane. I hated the project at first and it was very detrimental for my mental health in some aspects but i came out of it feeling so good about my skills.”

“I think after you finish gitlet, your confidence improves a lot (I didn't think i would ever be able to code something like this) but initially, it was a painful experience. Once you get the hang of serializing stuff the design is probably the hardest part.”

Some Gitlet Comments

“a project ABOUT git seems like a really shitty project when you first read the 60 page long spec (especially from someone who didn't know how to use git)... but the creativity and building that goes into it makes a really rewarding experience. i loved it!! “

“I just want to say that coming into college, I had never coded before. I was super daunted by just looking at the Gitlet spec when it was assigned and kept on pushing it off, but now I am proudly able to say that I have finished the project and it truly has given me a lot of confidence in my ability to code and think through lots of things at once.”

What Would You Cut From Gitlet?

What Would You Cut From Gitlet?

Workload Overall

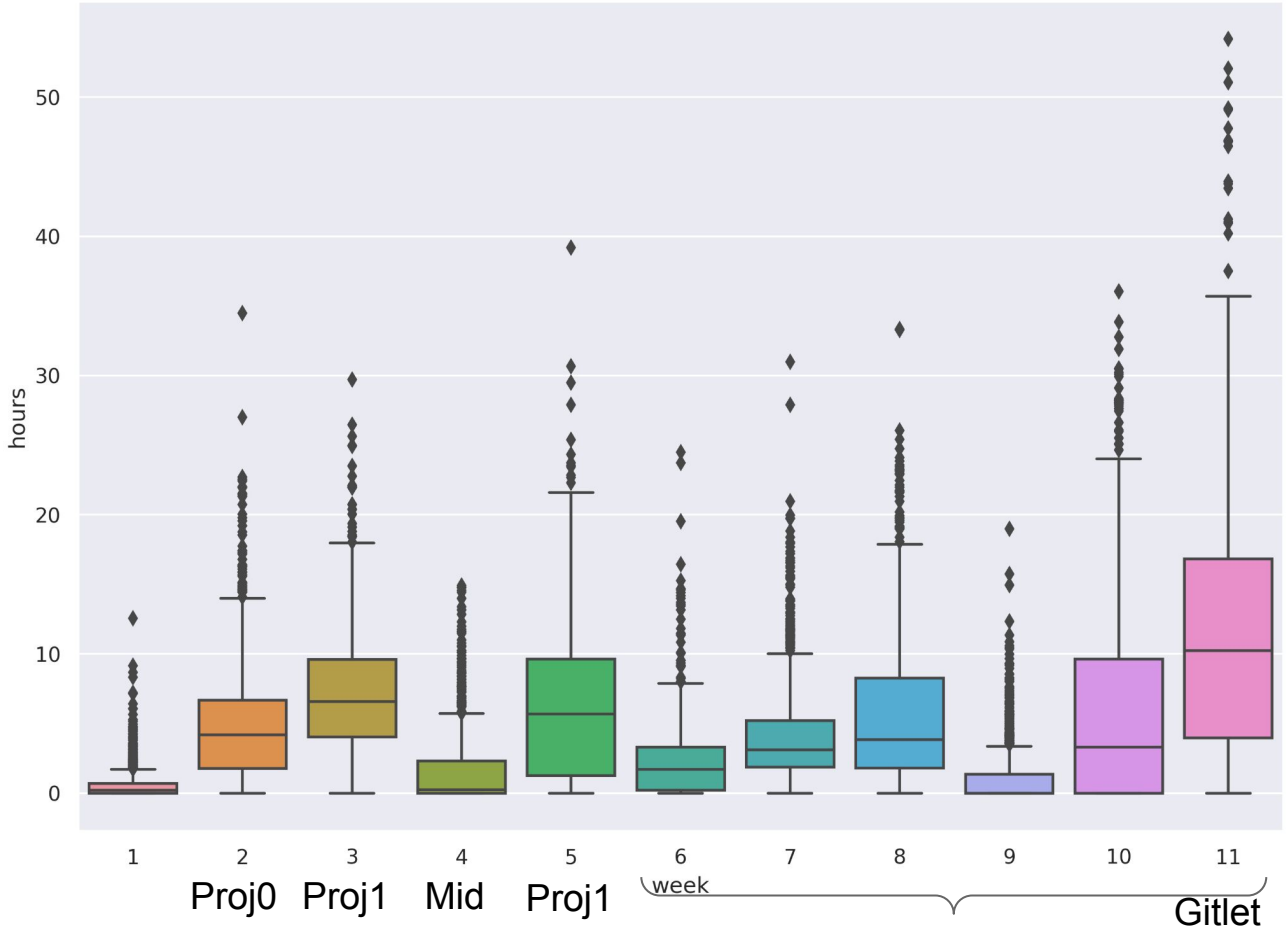
Workload in Theory

Berkeley recommends that a 4 unit class takes 12 hours per week. For us, that's:

- 1 hour discussion.
- 2 hours lab.
- 3 hours lecture.
- 6 hours everything else:
  - Check-in exercises / studying.
  - Surveys.
  - HW/projects.

Total time spent programming should be ~6 hours per week [~2 lab, ~4 everything else] on average.

Time Spent Per Week in IntelliJ



Statistics:

- Mean: 4.6 hours.
- Median: 4.3 hours.
- 25%: 3.2 hours.
- 75%: 6 hours.

Seems like we're actually doing OK? (except Gitlet)

The Role of Community

Formal and Informal Networks of Support

In a course, students benefit from both formal and informal networks of support.

- Formal: Office hours, Ed, Discussion, Lab, Project 3 Partner, etc.
- Informal: People you randomly talk to in a dorm, on the student run discord, etc.

I believe the ability to tap into these networks has a huge impact on success.

Reflection on the Project 3 Partnerships

What are some things you learned while working with a partner during project 3?

- Had no partner.
- If youw ant something done, do it yourself (bummer).
- We have different strengths.
- Other people don’t stay up until 4 AM.
- People programming differently.
- Working in groups is hard.
- It is often hard to explain waht you’re trying to do.
- Sticking to a schedule is good.
- Going over design with someone is good.
- People program at different paces.
- Programming in front of someone is harder than by yourself.
-

Reflection on the Project 3 Partnerships

What are some things you learned while working with a partner during project 3?

- Arguing for 3 hoursl eads to ocean ideas
- Feel compelled to share thoughts as thinking.
- Bett3er to have one commanding, other typing.
- Commenting is necessary.
- Hard to read som,eone els’e scode (also your own code one month alter)
- Neither of us know how to use uintelliJ
- My spelling is bad.
- Merge conflicts are tricky.
-

What makes a good teammate? (TA Tips)

- Communication.
  - Be upfront about what you can and cannot do.
  - Don’t ghost your team!
- Respecting other people’s ideas/contributions.
  - Avoid talking over other people.
  - Make compromises on your implementation.
- Acknowledgment and support.
  - Saying “thank you” when someone else does good work can improve morale.
- Managing implicit bias.
  - We all have implicit biases. Important to be aware of them and make sure we don’t let them cause harm.
- Standing up for other team members when necessary.

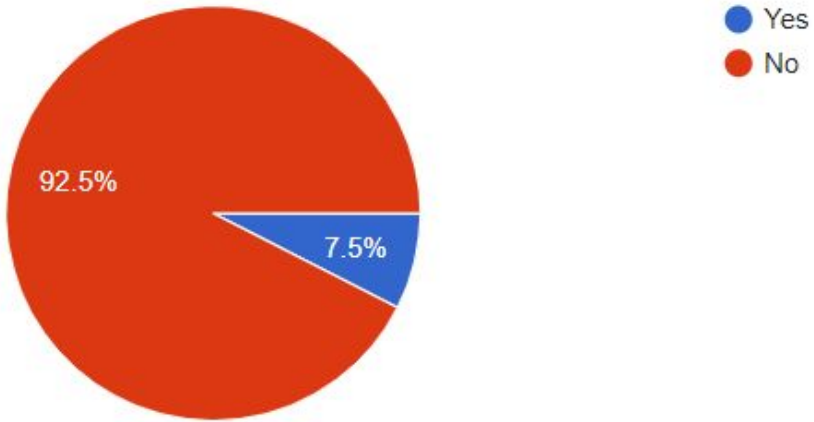
Harmful Behavior

7.5% of students reported seeing hurtful behavior or microaggressions.

- FWIW I don't like the term "microaggression" as it implies malice.
- An emerging term is "subtle acts of exclusion." (Link)

Have you ever witnessed any type of hurtful behavior or microaggression in a Berkeley CS class?

1,197 responses



Examples of Subtle Acts of Exclusion from the Survey

- “One time I went to OH and the tutor told me "Oh wow, you actually understand what I'm saying." Not too sure how to feel about that since I'm a woman.”
- “I have not experienced anything super explicit or egregious, but it is no secret that CS classes and spaces can be a bit of a boys' club. I have felt talked down to or underestimated as a girl in CS spaces, usually from male peers who feel they know more than me or are more qualified.”
- “A lot of boasting... like a lot of boasting by some people (this was particularly bad last semester in 16a since there was a group of students who would finish the lab in 20 minutes and then leave)”

Examples of Subtle Acts of Exclusion from the Survey

- “There was a person in the CS61B discord making inappropriate comments and jokes about harassment in EECS after a person shared their personal story about being harassed and stalked by a CS GSI. They made jokes about how it is okay or normal for people to become GSIs to "get girls," and said "gotta respect your GSIs." It felt like they were making a joke out of people's sexual harassment stories. After being called out for being insensitive, they replied that people should take be able to jokes and called the person who called them out "ignorant."”
- “In [upper Div CS class] last semester, I was in a discussion with a female TA (PhD student) and male students would continuously ask if she was sure about her answer when she answered their questions and seemed to always doubt that she was answering correctly.”
- “it was weird but other people called it out so i was glad to see that”

… BTW, Why Are We Talking About This?

You may be wondering why we're talking about this in a highly technical advanced data structures class.

Two reasons:

- We're all citizens of the university. IMO, it is incumbent on us to do our part to make our community a good place for our fellow students.
  - Who better to discuss with this than me, your professor?
- As my GSI Allyson puts it "A good programmer is not just someone who can solve any coding problem or ace any test. A good programmer is also someone that others want to work with!

Quick Thought Experiment

You're in a project group with two male students, John and Michael, and a female student, Maria. You notice throughout the conversation that John directs all of his questions to Michael. When Maria suggests an idea for how to implement some task, John cuts her off and says "Eh, I don't think that would work." However, a few minutes later, when Michael suggests the same thing, John says "Yeah, let's try that!"

What are some actions you could take in this situation?

… And We're Done!

Next Friday we'll do more reflecting. Hope you found that fun!